A Privacy-Preserving Graph Neural Network for Network Intrusion Detection

Xinjun Pei, Xiaoheng Deng, *Senior Member, IEEE,* Shengwei Tian, Ping Jiang, Yunlong Zhao, and Kaiping Xue, *Senior Member, IEEE,*

***Abstract*—With the ever-growing attention on communication security, machine learning-based network intrusion detection system (NIDS) is widely utilized to meet different security requirements. However, most of the existing methods manually extract or learn features from raw traffic, which is usually expensive, complicated, and time-consuming. Moreover, this also brings unprecedented challenges for preserving users' privacy in the communication process, making it difficult for existing solutions to be deployed in practice due to the privacy requirements from legal policies. This paper proposes a privacy-preserving graph neural network (named NIGNN) for NIDS, which can encode the local structure and traffic features. To address the privacy issues pertaining to the application of graph representation learning, we design a privacy message-passing mechanism with formal privacy guarantees, in which sensitive information potentially contained in graph vertices will be kept private. Specifically, we design a privacy-enhancement graph representation that introduces a degree-sensitive item in vertex-based aggregation to reduce noise. Our theoretical analysis shows that NIGNN can provide a provable privacy guarantee. Extensive experiments demonstrate NIGNN's performance in maintaining a sound privacy-accuracy trade-off.**

***Index Terms*—Network Intrusion Detection, Privacy-Preserving, Supervised Learning, Differential Privacy, Graph Neural Networks.**

I. Introduction

In recent years, network intrusion detection has attracted widespread attentions from academia and industry. While users have greatly benefited from the convenience brought by the Internet, network intrusion incidents have undoubtedly increased public concerns about security [1] [2]. The evolution of network intrusion attacks has prompted researchers to continuously develop new network intrusion prevention systems (NIDS) to identify the network intrusion activities.

With the rapid development of neural networks, deep learning (DL)-based NIDS methods have become state-of-the-art solutions. Such methods often extract traffic features manually and feed them into the well-designed model to obtain classification results. Existing DL-based NIDS methods extract various traffic features (e.g., message type sequences, packet lengths, and raw byte sequences) as input data to feed into the classification model. However, in the real world, network traffic data often exists in the form of graphs, which contain the long-term evolution and dynamic changes of network traffic flow. Representing network traffic as a graph provides a more effective means of capturing dependencies in a traffic network, thereby maximizing data integrity. Unfortunately, most existing DL-based NIDS methods only focus on extracting and learning statistical features of network packets, while ignoring the structural information of the network traffic data [3] [4]. Due to the lack of utilization of the structure and topology of network flow data, they can result in the neglect of valuable information derived from packet relationships in the flow data. As a result, these systems may fall short in detecting complex network attacks. Therefore, we introduce the concept of mapping network traffic as a non-Euclidean graph with packet relationships to maximize data integrity. A general traffic topology graph is used to describe the traffic network instead of the representation of Euclidean space data in traditional DL-based methods, thereby better capturing packet relationships.

Graph Neural Networks, as a subfield of deep learning, have demonstrated remarkable success across domains, such as computer vision, natural language processing, and social network analysis. Their adaptability to diverse non-Euclidean graph structures makes them ideal for analyzing complex network traffic patterns. To leverage the topological and attribute information of network traffic, some studies have applied graph neural networks (GNNs) to NIDS. For instance, Zheng *et al.* [3] introduced a GCN-based method for network traffic classification, which combines traffic trace graphs with statistical features to achieve high classification accuracy even with very few labeled data. Zhu *et al.* [4] proposed a Darknet Graph Neural Network (DGNN) for darknet traffic classification, which can effectively curb malicious darknet activities. However, many existing graph-based NIDS methods ignore privacy concerns. Traffic data are vulnerable to various privacy graph attacks. To overcome this challenge, we introduced a privacy-preserving graph neural network (named NIGNN) for NIDS. We extend the differential privacy (DP) method in the context of NIGNN to effectively preserve data privacy.

Existing studies employ DP algorithms [5] [6] [7] [8] to train deep learning models to defend against privacy inference attacks, such as private inference attacks and membership

Corresponding author: Xiaoheng Deng.

Xinjun Pei, Xiaoheng Deng, Ping Jiang, and Yunlong Zhao are with the School of Electronic Information, Central South University, Changsha, 410083, China, and are also with the Shenzhen Research Institute, Central South University, Shenzhen, 518000, China (E-mail: pei_xinjun@163.com, dxh@csu.edu.cn, pjiang@csu.edu.cn, 214711102@csu.edu.cn).

Shengwei Tian is with the School of Software, Xinjiang University, Wulumuqi, 830001, China (E-mail:tianshengwei@163.com).

Kaiping Xue is with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China (E-mail:kpxue@ustc.edu.cn).

inference attacks. This is typically done by adding DP noise to the gradients of the model during training, or by training the model using a noisy loss function. For example, Pan *et al.* [5] proposed a regression model based on an adaptive DP mechanism, which dynamically allocates privacy budgets and adds noise to the objective function. Furthermore, the work in [6] developed a private-preserving deep learning, which applies DP to a stochastic gradient descent (SGD) algorithm. They performed a refined analysis of privacy costs by tracking privacy loss. Similarly, Jaewoo *et al.* [7] proposed a DP-based SGD solution, which can be improved by carefully allocating privacy budgets for each iteration. Additionally, the study in [8] introduced a private LSTM language model to provide a user-level privacy guarantee. Although these methods introduce private-preserving DL models to protect user sensitive data, they may not be suitable for many graph learning-based scenarios, such as those based on GNN models. There are still theoretical and practical challenges. In this paper, we extend the application of DP to node classification tasks with GNNs.

Nonetheless, training a GNN model [9] under the constraint of DP is highly challenging than other privacy deep learning models due to the particularity of graphs. In GNNs, vertices are interconnected through edges. The GNN model updates vertex representations via a messaging framework, involving multiple exchanges of hidden representations between adjacent vertices [10]. Consequently, applying DP to GNN models may incur high noise [10], potentially compromising the data utility. In addition, the influence of neighbors on the aggregation is not uniform, which limits the ability of GNNs to learn vertex representations. To overcome this challenge, this paper designs a degree-sensitive privacy-enhancement measure for GNN aggregation and updates, which exerts different effects on vertices by incorporating contributions from numerous neighboring vertices. This approach effectively reduces the impact of DP noise, allowing for a more precise representation of vertices. The involvement of more neighbors will offer significant advantages in mitigating the adverse effects of DP noise. In this case, the utility of the model can be preserved without consuming too much privacy budget.

Based on the above ideas, we propose a privacy-preserving NIGNN for network intrusion detection, which transforms network traffic data into graph structures wherein vertices represent the traffic, and edges represent the IP hosts. Shaping the network-wide traffic states as a graph can describe the network traffic distribution in a heterogeneous spatial space. This design provides an intuitive description of the network-wide traffic state, allowing graph analysis techniques to detect network intrusion activities effectively. Besides, we train the NIGNN with a privacy graph convolution layer to prevent adversaries from inferring sensitive information, which uses a correlation coefficient perception (CCP)-based DP mechanism to enforce privacy preservation to the GNN model. To reduce the noise introduced by the DP mechanism, we design a degree-sensitive privacy-enhancement measure in the aggregation and update of GNN, which incorporates contributions from numerous neighboring nodes to generate a more accurate representation of the underlying data. In short, the proposed NIGNN ensures the practicality of the model while avoiding potential privacy leakages. To the best of our knowledge, we are the first to investigate the problem of privacy-preserving GNN model in NIDS, which prevents network intrusions while preserving user privacy. The main contributions of this paper are summarized as follows:

- We propose a novel graph construction method for network intrusion detection, which shapes the network-wide traffic states into a graphical representation describing network traffic distribution in a heterogeneous space. This gives sophisticated feature representations using graph structures. This enables excellent potentials in modeling sophisticated feature representations and graph structures.
- We develop a privacy-preserving GNN to identify network intrusions, which utilizes a CCP-based DP mechanism to enforce privacy preservation to the GNN models. We design a degree-sensitive privacy-enhancement measure to exert different effects on vertices, which can reduce the noise introduced by the DP mechanism while effectively learning an accurate model.
- Extensive experiments carried out on real-world datasets demonstrate the effectiveness of NIGNN in identifying network intrusions. The proposed NIGNN can achieve a high detection rate while providing a rigorous privacy guarantee.

The rest of the paper is structured as follows. Section II describes the related works. Section III briefly introduces some preliminaries. Section V gives the privacy-preserving NIGNN in details. Section VI evaluates the proposed NIGNN. Finally, we conclude the paper in Section VII.

II. Related Work

In recent years, deep learning (DL) methods have been applied in NIDS with unprecedented detection performance. In general, these methods follow a prevailing paradigm: the features extracted from raw traffic are first mapped into dense vector representations (e.g., word embeddings), and then concatenated together to input into a deep learning model to extract and learn high-order latent information. The work in [11] tested a variety of DL models and demonstrated the effectiveness of shallow neural network architectures in detecting network intrusions. Moreover, Liu *et al.* [12] presented a traffic obfuscation framework to prevent traffic analysis attacks. Das *et al.* [13] proposed a natural language processing to convert HTTP requests into vectors. Then, the learning model is used to detect anomalous traffic. Moreover, Liu *et al.* [14] designed a utility-optimal differentially private mechanism in cognitive radio networks, which provides real-time differential location privacy. Hsu *et al.* [15] used a deep reinforcement learning model to reflect traffic behaviors, which has the self-updating capability. However, these methods cannot directly operate on graph structure data [10], thereby ignoring the spatial structure information of traffic network.

The proliferation of network intrusions has presented unprecedented challenges in preserving users' privacy. While Internet anonymity dissociates an individual's identity from their network activities [16], any collection of data for NIDS may jeopardize the anonymity of users [17], thereby putting their privacy at risk. Sgaglione *et al.* [18] proposed a signature-based intrusion detection system that employs homomorphic encryption to enhance the security of sensitive data. However, using homomorphic encryption to encrypt data is computationally expensive, which affects the performance of the whole system. To reduce the computational overhead, Mokry *et al.* [19] proposed a privacy-preserving clustering algorithm for an intrusion detection system. This algorithm employs lightweight cryptographic techniques (mainly additive secret sharing) to protect the privacy of intrusion alert data. Additionally, Alazab *et al.* [20] introduced a federated learning (FL)-based intrusion detection system that creates a global detection model without sharing private data, where each entity trains its local model using its private data and then shares the model updates (gradients) with a central server. Similarly, Jin *et al.* [21] proposed an evolvable system architecture for an intrusion detection system, which utilizes a federated incremental learning method to aggregate the knowledge from diverse local models and incrementally update the FL model. However, the effectiveness of federated learning relies on the collaboration of a large number of devices. This collaboration can introduce a lot of communication overhead. Although these methods introduce privacy-preserving methods to protect sensitive intrusion data, they may not be suitable for GNN-based NIDS. In this paper, we extend the application of the differential privacy method to NIDS, which can not only identify network intrusion but also protect the privacy of intrusion data.

Recent GNN is considered an emerging research area and has been successfully applied across various fields, such as action recognition [22], click-through rate (CTR) prediction [23], and few-shot learning [24]. The ability to model irregular graph data is crucial for the GNN models, and many variants have been proposed, such as graph convolutional networks [10], graph attention networks [25], GraphSAGE [26], and so on. These methods recursively aggregate and transform neighborhood information to update vertex representations. As a result, the graph structure is encoded into the neural network to improve classification performance. As the deployment of these models becomes more widespread, there are concerns about graph privacy. However, due to the relational characteristics of graphs, training a privacy-preserving GNN model is more challenging than other privacy deep learning models. To address these challenges, researchers are turning their attention to the integration of privacy measures (e.g., differential privacy) into GNNs. There are a few attempts to provide privacy protection in the field of graph-based learning algorithms. Zhang *et al.* [27] reviewed various privacy attacks and privacy-preserving techniques in the graph domain. In [28], the authors categorized representative trustworthy GNN algorithms from a computational perspective. For instance, Wang *et al.* [29] proposed a privacy-preserving GNN for cloud environments that supports secure GNN training and inference by encrypting graph data using lightweight encryption techniques. In [30], the authors extended the privacy-preserving GNN in a federated learn-based recommendation system. They applied local differential privacy to local gradients to protect the privacy of user-item graphs. Another study by Miao *et al.* [31] introduced a privacy-preserving collaborative GNN for distributed graph databases, which utilizes a cluster-based DP algorithm to reduce model degradation. Similarly, Zhang *et al.* [32] proposed a distributed GCN framework that uses a subgraph sampling method to reduce communication and memory overhead. In another study, Bhaila *et al.* [33] studied the application of randomization mechanisms in high-dimensional feature settings, and utilized frequency estimates of graph clusters to supervise the training procedure at a sub-graph level. Sajadmanesh *et al.* [34] proposed a locally private graph neural network (LPGNN). They designed a multi-bit-based local differential privacy method to protect node privacy. Following this work [34], Du *et al.* [35] proposed an evaluation method to characterize the trade-off between utility and privacy for LPGNN. Furthermore, Lin *et al.* [36] utilized the multi-bit algorithm [34] to perturb node features, and employed a random response algorithm to perturb the graph structure. However, introducing too much noise can reduce data utility and compromise model quality. Although these methods introduce privacy-preserving GNN models to protect user sensitive information, there are relatively few studies on privacy-preserving GNN-based NIDS. There are still theoretical and practical challenges. Our work is inspired by recent advances in privacy-preserving techniques and graph representation learning. In this paper, we proposed a privacy-preserving GNN (named NIGNN) for NIDS. The main goal is to protect the privacy of network traffic data while leveraging the predictive power of GNNs for the identification of network intrusions.

III. Preliminaries

A traffic network consists of a set of IP hosts. Each pair of IP hosts may communicate with each other. This paper treats the traffic network as a directed graph, denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$ where $\mathcal{V}$ denotes the set of N vertices, and $\mathcal{E}$ denotes the set of M edges. We consider that the network intrusion detection can be transformed to a specific type of node classification task. We denote $X \in \mathbb{R}^{|\mathcal{V}| \times d}$ as a feature matrix. Each vertex $v_i \in \mathcal{V}$ has a d-dimensional feature vector $v_i = \mathbf{x}_i = \{x_{i1}, x_{i2}, ..., x_{id}\}$ with a corresponding label $y_i \in \{0, 1\}$. Formally, a labeled example is a tuple $(\mathbf{x}_i, y_i) \in \mathcal{G}$: a d-dimensional feature vector $\mathbf{x}_i$ with a label y_i. This task is to construct a GCN from $\mathcal{G}$ that enables us to output the prediction $\hat{y}_i$ for every vertex. $\hat{y}$ is used to estimate the probability of network intrusion appearing. Specifically, to prevent traffic analysis attacks, the proposed NIGNN is trained under the constraint of differential privacy. Next, we briefly introduce the definitions of DP and GNN.

A. Differential Privacy

Differential privacy is a privacy definition specifically designed for the problem of privacy-preserving data analysis.

TABLE I: Frequently used notations.

Notation	Description
ϵ	Privacy budget
Δ_f	Sensitivity
$\mathcal{D}$, $\mathcal{D}'$	Any two neighboring dataset
$\mathcal{A}$, X	The adjacency matrix and feature matrix
d	Dimension of feature
γ	Traffic state
h	Hidden neurons
ω	The parameter vector of the learning model
W	Weight matrix
r	Spearman correlation
β	Privacy budget ratio
thr	Threshold parameter
ρ	Control parameter

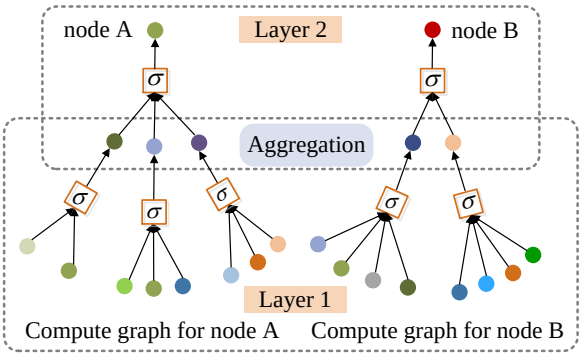


Fig. 1: Details of message passing.

The key idea behind DP [6] is that it does not permit discering of any individual's record. In other words, no matter what auxiliary information is available, the adversaries cannot infer the sensitive informantion of training data. To protect the privacy, we formally define DP as follows:

Definition 1 (ϵ-DP). A randomized algorithm $\mathcal{A}: \mathcal{D} \rightarrow \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies ϵ-DP, if for any two neighboring databases $\mathcal{D}$, $\mathcal{D}'$ differing on at most one tuple, and for all possible outputs $O \subseteq \mathcal{R}(\mathcal{A})$, we have:

$$\Pr[\mathcal{A}(\mathcal{D}) \in O] \leq e^{\varepsilon} \Pr\left[\mathcal{A}\left(\mathcal{D}'\right) \in O\right].$$

The privacy budget ϵ, as a metric of privacy loss, controls the privacy-utility trade-off. A smaller value of ϵ indicates a higher privacy guarantee but more reduced the data utility. Formally, we define the l_1-sensitivity as below.

Definition 2 (l_1-sensitivity). For any function $f: X \rightarrow \mathbb{R}^d$, the l_1-sensitivity of f is:

$$\Delta_f = \max_{\mathcal{D},\mathcal{D}'} \|f\left(\mathcal{D}\right) - f\left(\mathcal{D}'\right)\|_1.$$

The Laplace noise mechanism [37] is a well-known DP method, which can be described as below.

Theorem 1. Laplace mechanism. *For any function f, the Laplace mechanism $\mathcal{A}_f(D) \triangleq f(D) + Lap\left(0, \Delta_f/\epsilon\right)$ satisfies ϵ-DP.*

Proof. Proof of Theorem 1 can be found in [38].

The Laplace mechanism perturbs the private value with a random noise drawn from a Laplace distribution $Lap\left(0, \Delta_f/\epsilon\right)$ with mean zero and variance Δ_f/ϵ. The noise scale is calibrated by the l_1-sensitivity Δ_f (divided by ϵ).

B. Graph Neural Networks

This subsection presents a GNN model that directly operates on graph-structured data. Fig 1 gives an example. The GNN can perform effective information propagation on the graph. Let $(\mathcal{A}, X)$ be a tuple, which can be used as the input. X denotes the feature matrix, and $\mathcal{A}$ denotes the adjacency matrix. For every vertex in the graph, the GNN aggregates the vectors of its adjacent neighbors to learn the vertex's hidden representation. This process is shown in Fig. 1. The graph structure and vertex features can be encoded by the neighborhood aggregation operation $\mathrm{Agg}()$ and update operation $\sigma()$. More formally, the hidden representation h_v^l of a vertex v can be described as follows.

$$\mathbf{h}_{\mathcal{N}(v)}^l = \mathrm{Agg}\left(\left\{\mathbf{h}_u^{l-1}, \forall u \in \mathcal{N}(v)\right\}\right), \tag{1}$$

$$h_v^l(\theta) = \sigma\left(\left(\mathbf{h}_v^{l-1} \oplus \mathbf{h}_{\mathcal{N}(v)}^l\right) W^l\right), \tag{2}$$

where l is the number of layers, $\oplus$ represents the merge operation, W represents the weight matrix, and $\mathcal{N}(v)$ represents a set of its adjacent neighbors. Specifically, the $\mathrm{Agg}()$ is an aggregate function with invariant permutation, such as max, sum, or mean. Each layer updates all its hidden representations $H^l = \{h_v^l\}_0^n$ by an activation function σ (such as Sigmoid and ReLU), where n represents the number of hidden neurons. It is worth mentioning that, the GCN takes X as input, i. e., $h_{\mathbf{x}_i}^1(\theta) = \sigma\left(\left(\mathbf{x}_i \oplus \mathrm{Agg}\left(\{\mathbf{x}_u, \forall u \in \mathcal{N}(\mathbf{x}_i)\}\right)\right) W^1\right)$. Let $f\left(\mathbf{x}_i, \boldsymbol{\omega}\right)$ denote the *cost function* and $\boldsymbol{\omega}^*$ represents the optimal model parameter. The objective of the GNN is minimize the given cost function $f\left(\mathbf{x}_i, \boldsymbol{\omega}\right)$ in order to find the optimal parameter $\boldsymbol{\omega}$. Then, we give the definition of $\boldsymbol{\omega}^*$ as follows.

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} \sum_{i=1}^{n} f\left(\mathbf{x}_i, \boldsymbol{\omega}\right). \tag{3}$$

IV. Threat Model

A. System Architecture

In our study, we explore a typical IoT network, where IoT nodes connect to the Internet via an access gateway, as depicted in Fig. 2. We present a comprehensive network intrusion detection framework to defend against network intrusions and traffic private analysis. This framework includes the Security Gateway, Security Server, and Security Service.

1) Security Gateway: serves as a local access gateway to the Internet, allowing IoT devices to connect via Ethernet and WiFi. Its primary function is to monitor and collect communication data from all IoT devices. All Security Gateways upload their network traffic data to the Security Server for training a large-scale intrusion detection system. Another function of the Security Gateway is traffic filtering, which utilizes the predefined rules or policies to inspect, analyze and filter incoming and outgoing traffic. The predefined rules can be configured to block network traffic based on various criteria, such as destination IP address, source IP address, protocols, and port numbers. The Security Gateway enforces the filtering rules to restrict malicious traffic and reduce

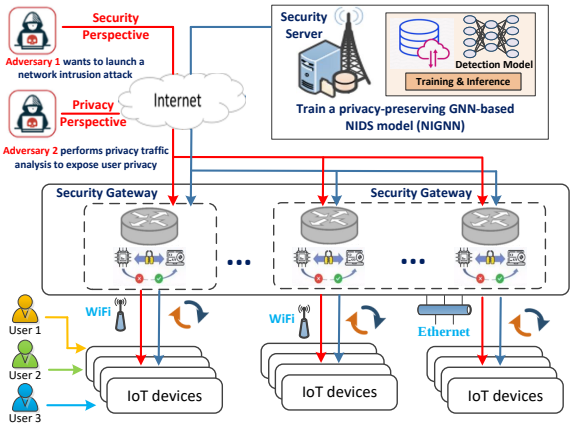


Fig. 2: Threat model.

the risk of network attacks. Furthermore, it monitors IoT device communications and detects abnormal communication behaviors caused by malicious IoT devices.

2) Security Server: We consider that Security Server has sufficient computational power to train a large-scale deep learning-based NIDS system. It is supported by the IoT security services provided by companies like Google and Microsoft. These services collect all network traffic data from security gateways to facilitate large-scale network intrusion detection and traffic behavior analysis.

3) Security Service: maintains a repository of NIDS to support Security Gateway to quickly query and identify abnormal traffic. Security Service periodically collects network traffic data to update the large-scale NIDS model provided by Security Server. Under cyber-attacks and other network anomalies, it also can support fast response and prevent the expansion of network attacks by blocking node communication.

B. System Model

In an IoT network, communication between IoT nodes occurs through the transmission of traffic, and each IoT node in the network has the ability of transmiting and forwarding traffic. Note that in most cases, traffic cannot be directly routed from the source IP node to the destination IP node. Instead, it typically necessitates traversal through one or more forwarding devices. However, due to the inherent limitations in the computing resources of IoT nodes, on-device monitoring is seldom feasible. In this context, Security Gateway switches stand out as the primary forwarding devices responsible for forwarding or discarding traffic, which serves as a vital component for security-related measures within IoT networks to identify and monitor traffic patterns, detect anomalies, and enforce security policies. The transmission process of all traffic within an IoT network can be mathematically expressed as follows:

$$SRC \rightarrow (N_{s_1}, R_{s_1}) \rightarrow (N_{s_2}, R_{s_2}) \cdots (N_{s_n}, R_{s_n}) \rightarrow DES, \quad (4)$$

where SRC indicates the source IP address of IP host H, DES indicates the destination IP address of IP host H, N_{s_i} indicates i-th IoT node, and R_{s_i} indicates the traffic processing rules associated with the IoT node N_{s_i}. Then, we establish the anomaly detection components in the Security Gateway for device communication monitoring and traffic capture, where each traffic is assigned to a specific class $y \in \{0, 1\}$. $y = 1$ indicates malicious traffic, and $y = 0$ indicates normal traffic. Then, the Security Gateway extracts statistic features from the raw network traffic to characterize the behavior of the traffic.

C. Adversary Model and Assumptions

We divide our threat model into two distinct classes of adversaries with differing visibility into the home network:

A1 - *Network intrusion attacks*: We consider an Adversary, referred to as "*Adversary* 1", who lacks knowledge of the home network topology but possesses the ability to launch attacks against IoT nodes from an external network. *Adversary 1* can launch a variety of network traffic attacks, such as backdoors, fuzzers, and DoS attacks, etc. The goal of *Adversary 1* is to execute network intrusion attacks that compromise as many IoT nodes as possible, thereby destabilizing device functionality and potentially infecting additional devices. Fig. 2 shows our threat model. For example, a DoS attack can deplete the resources of one or more servers, or block specific links within a data center. Ultimately, these coordinated attacks overwhelm the targeted servers with an excessive amount of network traffic.

A2 - *Privacy inference attacks based on traffic analysis*: Within the context of deep learning-based NIDS, the processing of the original network traffic leads to the generation of sequences of traffic features. These sequences encompass various attributes, such as traffic packets, traffic bytes, traffic duration, protocol types, TCP flags, etc. To establish a robust deep learning-based NIDS, the traffic features are typically transmitted to a Security Server for training the deep learning model. However, such communication raises privacy concerns, as traffic features contain a lot of important information, which could reveal private information about the activities of a home's occupants. For example, traffic rates from the indoor security camera in a smart home reveal whether the user is present at home when the camera detects movement, and traffic rates from the switch reveal when the IoT device is turned on or off. For example, a member inference attack [39] can infer whether a data record belongs to the training dataset of the target model. Suppose the training data is collected within a home network. When a network observer (referred to as "*Adversary* 2") determines a sub-training set using the member inference attack, *Adversary* 2 can reasonably infer a user's home activities from the sequence of traffic features of the sub-training set through traffic analysis.

D. Design Choices

This paper presents a novel privacy-preserving NIGNN for network intrusion detection, which shapes the network-wide traffic states into a graph. This method uses graph analysis method to detect network intrusion activities, effectively weakening the network intrusion attacks A1. Despite the

importance of privacy in traffic analysis, few efforts have been devoted to studying privacy inference attacks in this context of deep learning NIDS. This paper reveals how attackers can infer user privacy through traffic analysis. We trained the GNN model under DP constraints to mitigate privacy inferences from traffic feature sequences (privacy inference attack A2). By integrating privacy-preserving measures into the design and development of graph mining-based NIDS systems, it is possible to achieve a balance between effective intrusion detection and preservation of individual privacy. To our best knowledge, in the field of NIDS, it is the first attempt that graph anaysis-based NIDS is proposed to prevent network intrusion while preserving user privacy.

V. Differential Privacy Graph Neural Networks for Intrusion Detection

In this section, a privacy-preserving NIGNN is proposed. We consider that network intrusion detection can be regarded as a specific type of node classification task. In Section V-A, we transform the network traffics as graph data that describes the transition between network-wide traffic states at consecutive time steps. Section III-B describes how NIGNN encodes graph structure and traffic features. Section V-B describes how to construct a privacy graph convolution layer in NIGNN that perturbs the model input. Finally, we propose a degree-sensitive privacy-enhancement mechanism to reduce the noise in Section V-C.

A. Graph-based Traffic Representation

A traffic network usually consists of a set of IP hosts. Network intrusion detection can be seen as a specic time series analysis problem, which aims at predicting future network attacks in the trafc network using historical data. In this study, we transform the time series analysis into a node classication problem. Subsequently, a network-wide trafc graph is constructed to capture the spatial structure of the trafc data, which represents trafc ows as vertices and hosts as edges. Based on the link homophily [3], trafc ows with public IP hosts exhibit similar application trends, which can be measured on the graph. For instance, in a peer-to-peer (P2P) application, hosts frequently connect with multiple collaborating hosts, and trafc ows between them are associated with the same application [3]. Equally, we can obtain the same conclusion in client-server applications. We then use the NIGNN to model the temporal and spatial properties of the graph. This relational view of network trafc treats the node classication problem as information dissemination over the network-wide trafc graph. In this part, we contract a network-wide traffic graph.

Fig. 3 (a) illustrates a general traffic network. There are 6 IP hosts labeled as $s_1, s_2, ..., s_6$, where each IP host has some traffic packets. For example, the IP host S_1 have send two traffic packets $\gamma_{t_1}^{s_1}$ and $\gamma_{t_3}^{s_1}$ and received one traffic packet $\gamma_{t_2}^{s_2}$. The packets timestamps are denoted as t_i, where i = 1..k, and k is the total number of packets or nodes. When we measure the traffic data collected by s-th IP host for a certain period of time, the collected data sequence can be characterized in the form of a time series, denoted by $\gamma^s = \{\gamma_1, \gamma_2, \ldots, \gamma_T\} \in \mathbb{R}^{T \times M}$. We extract characteristics for each traffic packet. The extracted set of features can now describe each traffic packet. For the s-th IP host S_i, its traffic packet γ_t^s at time t is described by several features, such as timestamps, inter-packet mean time, and flow duration. For simplicity, we only use directed connections to construct the edges of the graph. Consequently, the traffic data sequence is transformed into the form of a network-wide traffic graph.

We are interested in utilizing the graph structure to analyze the trafc. In our case, the traffic data is transformed to a graph representation. We demonstrate the mapping of a network traffic flow into a graph-structured representation in which each packet is assigned to a node, and packet relations are encapsulated in edges with the chronological relationship serving as the edge direction. Fig. 3 (b) illustrates a general graph-structured network traffic flow representation. The details of how a network traffic flow is mapped to each graph entity are presented as follows.

- In Fig. 3 (a), the traffic $\gamma_{t_1}^{s_1}$ and $\gamma_{t_3}^{s_1}$ have the common host s_1. In Fig. 3 (b), there exists a directed edge between $\gamma_{t_1}^{s_1}$ and $\gamma_{t_3}^{s_1}$.
- We regard the receiving and sending traffic of a host as two vertices in the graph, and use a directed connection to construct the edge between the two vertices, e.g. $\gamma_{t_1}^{s_1}$ is linked to $\gamma_{t_2}^{s_2}$ in Fig. 3 (b).
- We set a threshold thr that designates the maximum time interval. Temporal information is the inter-arrival time between packets and can be calculated by $t_r \ t_s$. In this scenario, s is the sender node index and r is the receiver node index. When the threshold thr is set to $1, |t_3 - t_1| > thr$. As shown in Fig. 3 (b), there is no edge between the two vertices $\gamma_{t_1}^{s_1}$ and $\gamma_{t_3}^{s_2}$.

For simplicity, all traffic data can form a traffic feature matrix $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where $X \in \mathbb{R}^{N \times M}$, and each element $\mathbf{x}_1$ represents a traffic data γ_t^s. Let $\mathcal{A} \in \mathbb{R}^{N \times N}$ be the adjacency matrix of the network-wide traffic graph. When the traffic i and j have a common IP host, $A_{i,j} = 1$, otherwise $A_{i,j} = 0$. After that, the network-wide traffic graph is constructed, which can be represented as $\mathcal{G} = (X, \mathcal{A})$.

B. Private Graph Convolution Layer with Correlation Coefficient Perception Noise

In our proposed NIGNN, we construct a private gaph convolution (PGC) layer that perturbs the input features. Fig 4 shows an overview of NIGNN architecture. To achieve DP, a baseline approach is to insert the same magnitude of noise (selected from an identical noise distribution $\frac{1}{|D|} Lap\left(0, \Delta_h/\epsilon_1\right)$) to the all input X. Intuitively, the baseline approach can work well. In practice, this assumption is not valid, and the utility of the model may be affected as the contribution of each feature may vary.

Moreover, we propose a correlation coefficient perception (CCP)-based DP mechanism, which relies on a weighted correlation coefficient matrix to inject different magnitudes of noise into the model input. Since features are dependent of each other (e.g., duration versus number of packets, flow

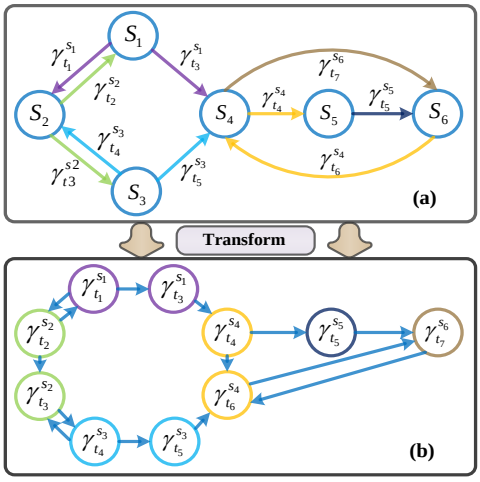


Fig. 3: Details of graph transformation.

byte rate versus flow packet rate, and minimum segment size versus minimum inter-arrival time of packets), we adopt CCP to quantify the extent of statistical dependence between two variables. In CCP, Spearman rank correlation coefficient is recommended for data with deviations or outliers [40]. In essence, rank correlation analysis is conducted for measuring both linear and general relationships between two variables. It determines whether one variable takes on a larger or smaller value concerning the other variable, although not necessarily in a linear manner [40].

Recall that feature matrix X consists of a set of traffic data $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$, where each row $\mathbf{x}_i$ contains d features $x_{i1}, x_{i2}, ..x_{id}$. Let $\mathbf{t}^{(p)} = \{x_{1p}, x_{2p}, ..., x_{np}\}$ and $\mathbf{t}^{(q)} = \{x_{1q}, x_{2q}, ..., x_{nq}\}$ be two column vectors. Let $\mathbf{t}'^{(p)} = \{x'_{1p}, x'_{2p}, ..., x'_{np}\}$ and $\mathbf{t}'^{(q)} = \{x'_{1q}, x'_{2q}, ..., x'_{nq}\}$ be the permutations of $\mathbf{t}^{(p)}$ and $\mathbf{t}^{(q)}$, respectively. Spearman correlation between $\mathbf{t}^{(p)}$ and $\mathbf{t}^{(q)}$ can be described as follows.

$$r_{(\mathbf{t}^{(p)},\mathbf{t}^{(q)})} = \frac{\operatorname{cov}\left(\mathbf{t}'^{(p)}, \mathbf{t}'^{(q)}\right)}{\widehat{\sigma}_{\mathbf{t}'^{(p)}} \widehat{\sigma}_{\mathbf{x}'^{(q)}}}, \tag{5}$$

where $\widehat{\sigma}_{\mathbf{t}'^{(p)}}$ and $\widehat{\sigma}_{\mathbf{t}'^{(q)}}$ denote the standard deviations of rank variables, and $\operatorname{cov}\left(\mathbf{t}'^{(p)}, \mathbf{t}'^{(q)}\right)$ represents the covariance of rank variables. Only if all n ranks are distinct integers, Spearman correlation can be computed by $r_{(\mathbf{t}^{(p)},\mathbf{t}^{(q)})} = 1 - \frac{6\sum_{i=1}^{n} df_i^2}{n(n^2-1)}$ [41] where $df_i = x'_{ip} - x'_{iq}$ is the difference between the two ranks.

Intuitively, the Spearman correlation between $\mathbf{t}^{(p)}$ and $\mathbf{t}^{(q)}$ is large (resp. small) when observations have a similar (resp. dissimilar) rank. For j-th input feature x_{ij}, we calculate the Spearman correlations between it and all other features (i.e., $r_j = \{r_{(\mathbf{t}^{(j)},\mathbf{t}^{(1)})}, r_{(\mathbf{t}^{(j)},\mathbf{t}^{(2)})}, ..., r_{(\mathbf{t}^{(j)},\mathbf{t}^{(d)})}\}$). As a result, we can denote a privacy budget ratio as $\beta_j = \frac{|\overline{r}_j|}{\frac{1}{d}\cdot\sum_{j=0}^{d}|\overline{r}_j|}$. For j-th input feature, the privacy budget can be denoted as $\epsilon_j = \beta_j \cdot \epsilon$.

As mentioned earlier, each traffic data $\mathbf{x}_i$ is represented by a d-dimensional vertex vector $\mathbf{x}_i$, where $\mathbf{x}_i = \{x_{i1}, x_{i2}, ..x_{id}\}$. Here, the PGC layer perturbs each vertex

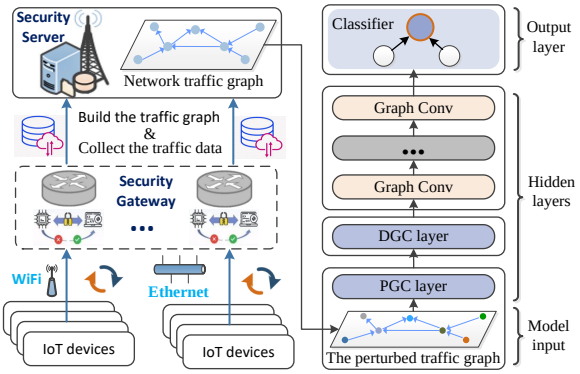


Fig. 4: An overview of NIGNN architecture.

feature x_{ij} of $\mathbf{x}_i$. For j-th input feature x_{ij} of x_i, we can have:

$$x'_{ij} = PGC_{\mathbf{x}_i}(x_{ij}) = x_{ij} + \frac{1}{|\mathcal{D}|} Lap\left(0, \Delta_f/\epsilon_j\right), \tag{6}$$

where x'_{ij} is the perturbed vertex feature. Specifically, Δ_f is set to $2\sum_f d$. Then, the perturbed vertex data $\mathbf{x}'_i$ can be represented by:

$$\begin{aligned} \mathbf{x}'_i &= \{x'_{i1}, x'_{i2}, ..., x'_{id}\}. \\ &= \{PGC_{\mathbf{x}_i}(x_{i1}), PGC_{\mathbf{x}_i}(x_{i2}), ..., PGC_{\mathbf{x}_i}(x_{id})\}. \end{aligned} \tag{7}$$

After that, the Security Gateway uploads the perturbed data $\mathbf{x}'_i$ to the Security Server to train a privacy-preserving GNN. On the Security Server side, we construct the PGC layer $\overline{PGC}_{\mathcal{D}}(W)$, and take the perturbed vertex features $\mathbf{x}'_i$ as input. The PGC layer $\overline{PGC}_{\mathcal{D}'}(W)$ consists of a set of hidden neurons $\bar{h}_{\mathcal{D}}(W)$:

$$\bar{h}_{\mathcal{D}'}(W) = \sum_{\mathbf{x'}_i \in \mathcal{D}'} (\mathbf{x'}_i \cdot W), \tag{8}$$

$$\overline{PGC}_{\mathcal{D}'}(W) = \left\{\bar{h}_{\mathcal{D}'}(W)\right\}_{\overline{h} \in \overline{PGC}}. \tag{9}$$

As explaining in Algorithm 1, we first determine the sensitivity Δ_f and the privacy budget ϵ_j (lines 2-4). Then, each vertex aggregates the information of its neighbors (lines 6-8) followed by a perturbation with Laplace noise (lines 9-10). Lastly, all hidden neurons are updated (lines 12-14). We give the bound of the sensitivity Δ_f as follows.

Lemma 1: Let $\mathcal{D}$ and $\mathcal{D}'$ be any two neighboring datasets. Let $\overline{PGC}_{\mathcal{D}}(W)$ and $\overline{PGC}_{\mathcal{D}'}(W)$ be the two PGC layers on $\mathcal{D}$ and $\mathcal{D}'$, respectively. We have:

$$\begin{cases} h_{\mathcal{D}}(W) = \sum_{\mathbf{x}_i \in \mathcal{D}} (\mathbf{x}_i \cdot W) \\ \overline{PGC}_{\mathcal{D}}(W) = \{h_{\mathcal{D}}(W)\}_{h \in \overline{PGC}} \end{cases}$$

$$\begin{cases} \bar{h}_{\mathcal{D}'}(W) = \sum_{\mathbf{x}'_i \in \mathcal{D}'} (\mathbf{x}'_i \cdot W) \\ \overline{PGC}_{\mathcal{D}'}(W) = \{\bar{h}_{\mathcal{D}'}(W)\}_{\overline{h} \in \overline{PGC}} \end{cases}$$

Then, we have the following inequality:

$$\begin{aligned} \Delta_f &= \sum_{h \in \overline{PGC}} \sum_{j=1}^{d} \left\| \sum_{\mathbf{x}_i \in \mathcal{D}} x_{ij} - \sum_{\mathbf{x}'_i \in \mathcal{D}'} x'_{ij} \right\|_1 \\ &\leq 2 \sum_{h \in PGC} d. \end{aligned}$$

Algorithm 1 Private Graph Convolution Layer with DP

Input: Original traffic data X, privacy budget ϵ
Output: The perturbed PGC layer $\overline{PGC}_{\mathcal{D}'}(W)$

Security Gateway side:
$\Delta_f = 2\sum_{h\in H_0} d$;
for each $0 \le j \le d$ **do**
  $\epsilon_j = \beta_j \cdot \epsilon_1$;
end for
for $\mathbf{x}_i \in \mathcal{D}, i \in [0,n]$ **do**
  for $x_{ij} \in \mathbf{x}_i, j \in [0,d]$ **do**
    Perturb the vertex feature:
    $x'_{ij} = PGC_{\mathbf{x}_i}(x_{ij}) = x_{ij} + \frac{1}{|\mathcal{D}|} Lap(0, \Delta_f/\epsilon_j)$;
  end for
  Obtain the perturbed vertex representation:
  $\mathbf{x}'_i = \{x'_{i1}, x'_{i2}, ..., x'_{id}\}$;
end for
Update $PGC'_{\mathbf{x}'_i}$ to the Security Server;
Security Server side:
Construct the PGC layer:
$\overline{h}_{\mathcal{D}'}(W) = \sum_{\mathbf{x}'_i \in \mathcal{D}'} (\mathbf{x}'_i \cdot W)$;
$\overline{PGC}_{\mathcal{D}'}(W) = \left\{\overline{h}_{\mathcal{D}'}(W)\right\}_{\overline{h}\in\overline{PGC}}$;
return $\overline{PGC}_{\mathcal{D}'}(W)$.

where d is the feature dimensions of $\mathbf{x}_i \in \mathcal{D}$.

Proof. Assume that $\mathcal{D}$ and $\mathcal{D}'$ differ in the last tuple. Let $\mathbf{x}_n$ ($\mathbf{x}'_n$) be the last tuple in $\mathcal{D}$ ($\mathcal{D}'$). We have that

$$\begin{aligned}\Delta_h &= \sum_{h\in\overline{PGC}} \sum_{j=1}^{d} \left\| \sum_{\mathbf{x}_i\in\mathcal{D}} x_{ij} - \sum_{\mathbf{x}'_i\in\mathcal{D}'} x'_{ij} \right\|_1 \\ &= \sum_{h\in\overline{PGC}} \sum_{j=1}^{d} \left\| x_{nj} - x'_{nj} \right\|_1 \\ &\le 2 \max_{\mathbf{x}_i\in\mathcal{D}} \sum_{h\in\overline{PGC}} \sum_{j=1}^{d} \left\| x_{ij} \right\|_1 .\end{aligned}$$

Since $\forall \mathbf{x}_i, j : x_{ij} \in [0,1]$, we have that: $\Delta_{\mathrm{h}} \le 2\sum_{h\in\overline{PGC}} d$.

Lemma 2: Algorithm 1 preserves ϵ-DP in the computation of $\overline{PGC}_{\mathcal{D}}(W)$.

Proof. According to the calculation of GCN, for each $h \in \overline{PGC}_{\mathcal{D}}$, h can be re-written as follows.

$$\overline{h}_{\mathcal{D}}(W) = \sum_{j=1}^{d} \left[\sum_{\mathbf{x}_i\in\mathcal{D}} \left(x_{ij} + \frac{1}{|\mathcal{D}|} Lap(0, \Delta_f/\epsilon_j) \right) W \right] = \sum_{j=1}^{d} \bar{\xi}^h_j W.$$

Then, we set Δ_h to $2\sum_{h\in H_0} d$, as shown in Algorithm 1. We have

$$\begin{aligned}\frac{\Pr\left(\overline{PGC}_{\mathcal{D}}(W)\right)}{\Pr\left(\overline{PGC}_{\mathcal{D}}(W)\right)} &= \frac{\prod_h \prod_{j=0}^{d} \exp\left(\frac{\epsilon_j \left\| \sum_{\mathbf{x}_i\in\mathcal{D}} x_{ij} - \bar{\xi}^h_j \right\|_1}{\Delta_h} \right)}{\prod_h \prod_{j=0}^{d} \exp\left(\frac{\epsilon_j \left\| \sum_{\mathbf{x}'_i\in\mathcal{D}'} x'_{ij} - \bar{\xi}^h_j \right\|_1}{\Delta_h} \right)} \\ &\le \prod_h \prod_{j=0}^{d} \exp\left(\frac{\epsilon_j}{\Delta_f} \left\| \sum_{\mathbf{x}_i\in\mathcal{D}} x_{ij} - \sum_{\mathbf{x}'_i\in\mathcal{D}'} x'_{ij} \right\|_1 \right) \\ &\le \prod_h \prod_{j=1}^{d} \exp\left(\frac{\epsilon_j}{\Delta_f} 2 \max_{\mathbf{x}_n\in\mathcal{D}} \left\| x_{nj} \right\|_1 \right) \\ &\le \prod_h \prod_{j=1}^{d} \exp\left(\frac{2\epsilon_j}{\Delta_f} \right) \le \prod_h \prod_{j=1}^{d} \exp\left(\frac{\epsilon}{\Delta_f} \cdot \frac{2\cdot|\overline{r}_j|}{\frac{1}{d}\sum_{j=1}^{d}|\overline{r}_j|} \right) \\ &\le \exp\left(\frac{\epsilon}{\Delta_f} \cdot 2 \sum_h d \left[\sum_{j=1}^{d} \frac{|\overline{r}_j|}{\sum_{j=1}^{d}|\overline{r}_j|} \right] \right) = \exp(\epsilon).\end{aligned}$$

Algorithm 2 Degree-Sensitive Graph Convolution Layer

Input: The perturbed PGC layer $\overline{PGC}$, adjacency matrix $\mathcal{A}$
Output: The degree-sensitive graph convolution layer $\overline{DGC}$

Obtain the degree of each vertex from the adjacency matrix $\mathcal{A}$:
$DegSet$ = *ObtainDegree*($\mathcal{A}$);
for $v \in \mathcal{V}, deg_v \in DegSet$ **do**
  Compute the degree item $D(v, deg)$ for each vertex:
  $d(v, deg) = \begin{cases} (deg_v)^{\rho}, deg_v > thr \\ \left(\frac{\epsilon}{deg_v}\right)^{\rho}, deg_v \le thr \end{cases}$
  Obtain the vertex representation:
  $\hat{\mathbf{h}}_v = D(v, deg) \cdot \mathbf{h}_v \oplus \text{Agg}\left(\left\{\mathbf{h}_u^{l-1}, \forall u \in \mathcal{N}(v)\right\}\right)$;
  $\hat{h}_v(W) = \sigma\left(\hat{\mathbf{h}}_v W\right)$;
end for
Construct the DGC layer:
$\overline{DGC} = \{\hat{h}_v(W)\}_{\hat{h}_v\in\overline{DGC}}$;
return $\overline{DGC}(W)$.

C. Degree-Sensitive Graph Convolution Layer

Recall that in the PGC layer, the DP mechanism is used to perturb every vertex features by adding Laplace noise $\frac{1}{|\mathcal{D}|} Lap(0, \Delta_h/\epsilon_j)$ into x_{ij}, and the level of noise introduced by each individual vertex $\mathbf{x}_i$ is controlled by the privacy budget ϵ. Hence, the presence of noise can significantly impact the accuracy and reliability of aggregated information. Considering the impact of noise introduced by the PGC layer on the aggregation process, we propose a degree-sensitive graph convolution (named DGC) layer $\overline{DGC}(\cdot)$, which manipulates the outputs of $\overline{DGC}(\cdot)$ by giving vertices with a very small number of neighbors relatively higher impacts on the model learning. The $\overline{DGC}(\cdot)$ is designed to alleviate the impact of the DP noise introduced by $\overline{PGC}(\cdot)$ and enhance the overall performance of the model. In the context of vertex-based aggregation, it is naturally quite sensitive to the number of neighbors. It is worth noting that the original GNN treats

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3417853

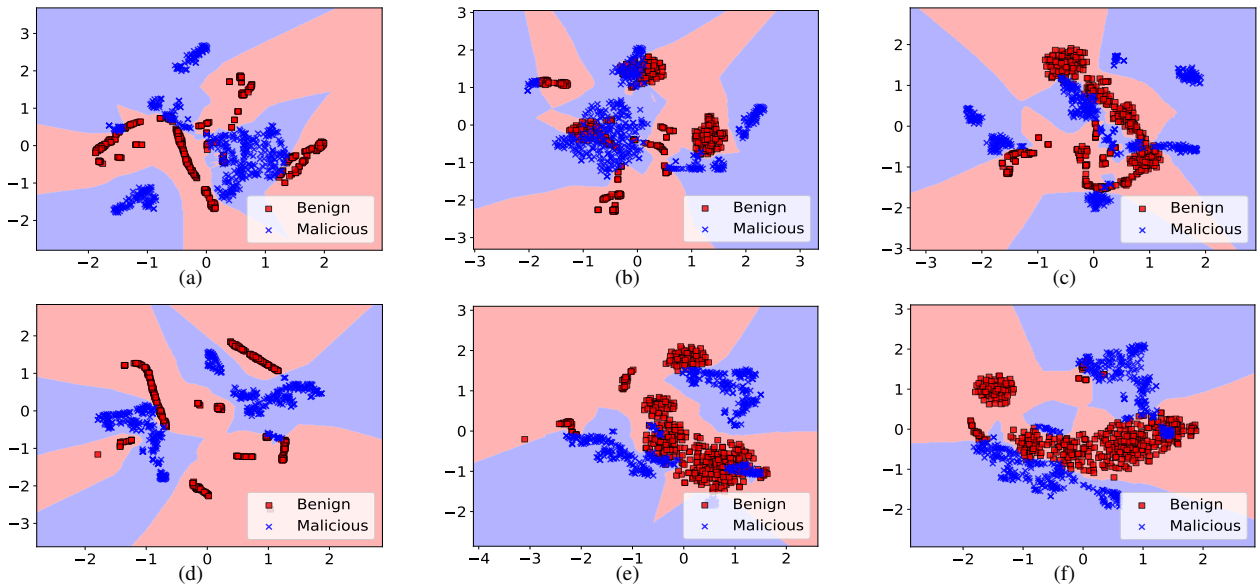


Fig. 5: The decision boundaries for the cluster of vertices with a small number of neigbors produced by the (a) raw network traffic graph, (b) $\overline{PGC}(\cdot)$ with $\rho = 0.8$ and $thr = 10$, and (c) $\overline{DGC}(\cdot)$ with $\rho = 0.8$ and $thr = 10$. The decision boundaries for the cluster of vertices with a large number of neigbors produced by the (d) raw network traffic graph, (e) $\overline{PGC}(\cdot)$ with $\rho = 0.8$ and $thr = 10$, and (f) $\overline{DGC}(\cdot)$ with $\rho = 0.8$ and $thr = 10$.

all the vertices equally. However, the influence of neighbors on the aggregation is not uniform.

In order to alleviate the impact of the DP noise, each vertex v is associated with a degree item $d(v, deg)$, where deg is the degree of v. The $\overline{DGC}(W)$ can act as a noise reducer by dynamically adjusting the update amplitude for each vertex based on its degree. The degree item $d(v, deg)$ indicates the vertex importance. Clearly, a vertex with more neighbors is more important. This is because the involvement of more neighbors can counteract the adverse effects of noise during the aggregation process, (i.e., $\mathrm{Agg}()$). In other words, the collective contribution from these vertices counteracts the noise introduced by each individual vertex by the PGC layer. The resulting aggregation reflects a more accurate representation of the underlying data. Therefore, vertices with more neighbors should assign smaller degree terms $d(v, deg)$, aiming to preserve the aggregation results obtained from the previous layer. This allows the model to learn with reduced noise, bringing the results closer to those of a noise-free model. Conversely, vertices with fewer neighbors are considered less important and are assigned higher values of the degree term $d(v, deg)$. This strategy encourages more substantial updates for vertices with fewer neighbors, facilitating the model's ability to move away from local optima and learn more effectively. In general, our method can balance the update amplitude based on the degree of each vertex. This ensures that both vertices with many or few neighbors contribute meaningfully to the learning process, enhancing the model's adaptability.

Formally, we define the degree term $d(v, deg)$ as follows.

$$d(v, deg) = \begin{cases} (deg_v)^{\rho}, deg_v > thr \\ \left(\frac{\epsilon}{deg_v}\right)^{\rho}, deg_v \leq thr \end{cases} \tag{10}$$

where thr is a threshold parameter, and $\rho \in [0, 1]$ is a control parameter.

Specifically, the DGC layer takes the output of the PGC layer as its input. When the privacy budget ϵ is small, the output of the PGC layer contains more noise. In contrast, when the privacy budget ϵ is large, the output of the PGC layer contains less noise. To accommodate different privacy scenarios, we incorporated the privacy budget parameter ϵ into the Eq. 10, allowing the model to dynamically adjust the update amplitude based on the level of privacy protection required. Intuitively, a smaller privacy budget ϵ (indicating more added noise) implies a need for a more cautious and conservative update amplitude (i.e., a smaller $d(v, deg)$). As a result, a smaller privacy budget ϵ should result in a smaller $d(v, deg)$, which ensures that the update amplitude aligns with the noise level.

In the DGC layer $\overline{DGC}(\cdot)$, the hidden representation h_v of a vertex v can be updated as follows.

$$\mathbf{h}_v = \mathbf{h}_v \oplus \ \mathrm{Agg}\ \left(\left\{\mathbf{h}_u^{l-1}, \forall u \in \mathcal{N}(v)\right\}\right), \tag{11}$$

$$\hat{\mathbf{h}}_v = d(v, deg) \cdot \mathbf{h}_v, \tag{12}$$

$$\hat{h}_v(W) = \sigma\left(\hat{\mathbf{h}}_v W\right), \tag{13}$$

where $\mathbf{h}_v$ represents the hidden representation of a vertex v, and $\mathbf{h}_u$ represents the hidden representation of a neighbor vertex u. $\mathcal{N}(v)$ represents the set of its adjacent neighbors, $\mathrm{Agg}()$ is an aggregate function, $\oplus$ represents the merge operation, and W represents the weight matrix. For a vertex v, Eq. 11 is used to aggregate information from its adjacent neighbors $\mathcal{N}(v)$. By introducing a degree item $d(v, deg)$, Eq. 12 acts as a noise reducer to alleviate the impact of noise introduced by the PGC layer. Following this, Eq. 13 updates

the hidden representation of a node v. Finally, the DGC layer employs Eq. 14 to update all hidden representations.

$$\overline{DGC}(W) = \{\hat{h}_v(W)\}_{\hat{h}_v \in \overline{DGC}}, \tag{14}$$

As a result, the DGC layer can capture and leverage relevant information from the graph structure while mitigating the adverse effects of noise, thereby enhancing the robustness and performance of the proposed NIGNN. Our method not only maintained the original network structure, but also strengthened the learning with regard to the vertex-based aggregation.

After building the DGC layer $\overline{DGC}(W)$, we stack the graph convolution layers $\{H^1, H^2, ..., H^l\}$ on the top of $\overline{DGC}(W)$. In Fig. 5, benign vertices and malicious vertices in the network traffic graph constructed in this paper are denoted by red and blue circles, respectively. This visualization serves to illustrate the impact of the number of neighbors. The first row shows vertices with a small number of neighbors, falling below the threshold thr. Conversely, the second row shows vertices with a large number of neighbors. Intuitively speaking, with the advantage in quantity, vertices with more neighbors tend to have clearer classification boundaries compared to those with fewer neighbors, leading to better classification results. The visual analysis presented in Fig. 5 sheds light on the crucial role played by neighbor number in the accuracy and robustness of the classification task.

Fig. 5 (a) and (d) focus on evaluating the effects of the raw network traffic graph without any additional modifications or enhancements, which can be served as a baseline. Fig. 5 (b) and (e) provide an overview of the effect of $\overline{PGC}(\cdot)$, which enforces privacy preservation to the standard GNN. Fig. 5 (c) and (f) illustrate the $\overline{DGC}(\cdot)$, which augments $\overline{PGC}(\cdot)$ by applying different impacts to vertex update process. A direct comparison of the results presented in Fig. 5 (b) and (e) with those in (c) and (f) allows for an assessment of the effectiveness of $\overline{DGC}(\cdot)$ in mitigating the adverse effects of noise and improving classification accuracy. By incorporating tailored vertex update strategies, $\overline{DGC}(\cdot)$ enables enhanced adaptation to the underlying network traffic patterns, ultimately enhancing the model's ability to accurately classify benign and malicious vertices.

D. The Correctness and Applicability of the NIGNN

This section summarizes the key steps of our NIGNN. Algorithm 1 presents the procedures of CCPM in the $\overline{PGC}$ layer, which enforces privacy preservation to the standard GNN. This procedure is independent of the number of training epochs. Δ_f depends on the dimensions of vertex features, but do not depend on the number of training epochs. According to Lemma 2, the $\overline{PGC}$ layer is ϵ-differentially privacy, and thus the computation of the $\overline{DGC}$ layer and l hidden layers $\{H^1, H^2..., H^l\}$ above $\overline{PGC}$ are differentially private because there is no additional information from the traffic data to be accessed.

VI. Performance Evaluation

In this section, we conduct a number of experiments using real-world datasets to evaluate the privacy-utility performance of NIGNN. We are interested in network intrusion detection where traffics are connected via directed edges, forming a graph. The task is to detect network intrusions.

TABLE II: Main datasets used in our evaluation studies.

Datasets	Tor-nonTor	DIDarknet	UNSW-NB
# $\lvert\mathcal{V}\rvert$	440,044	66,615	141,530
# $\lvert\mathcal{E}\rvert$	4,389,183	1,756,939	2,326,182
Avg. Deg.	8.99	25.67	17.13
Training size	265,546	40,199	85,324
Validation size	87,359	13,066	28,045
Test size	87,139	13,350	28,161

A. Datasets and Evaluation Metrics

1) Datasets: In our experiment, we evaluated NIGNN on three benchmark datasets. The DIDarknet [42], Tor-nonTor [43], and UNSW-NB [44] datasets are widely used in NIDS. The DIDarknet dataset [42] is an open-source repository that includes malicious traffic from the darknet and corresponding benign traffic from various sources such as Chat, Email, Browsing, etc. It also reflects how network nodes act in space, and how each node contacts with each other within a certain time interval. Note that the latest trend of using protocols (like VPN/Non-VPN) to encrypt and disguise Internet traffic makes network traffic classification an open challenge. In light of this, we also evaluated our method on the Tor-nonTor dataset [43], in which each Tor traffic includes a set of time-based features. The UNSW-NB dataset [44] mixes normal real-world network traffic with synthesized cyber-attack activity traffic, in which attacks have been categorized into nine types, such as Backdoors, Dos, Worms, etc. Table II provides details of these datasets.

2) Evaluation Metrics: In our experiments, we evaluate the performance of the privacy-preserving NIGNN based on some common performance evaluation metrics in machine learning, including accuracy and ROC. The primary objective of this paper is to achieve a high detection rate for network intrusion detection. Specifically, we shuffled the examples and trained on 60% of the data, validated on 20% of the data, and tested on the rest 20% as listed in Table II.

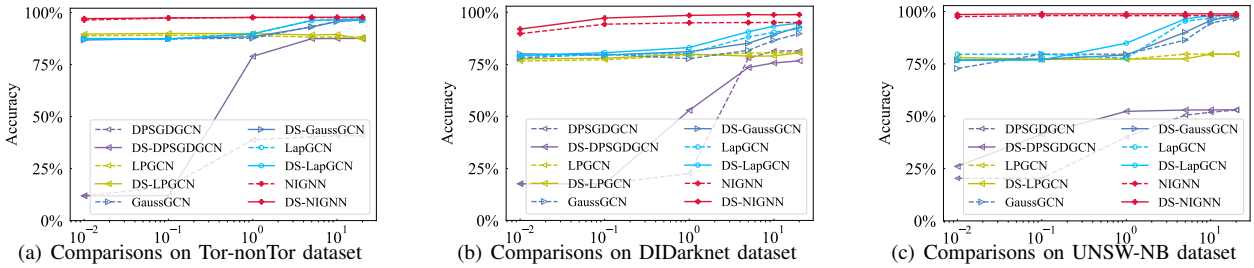
B. Comparative Classification Performance

In this experiment, we compare NIGNN with other private-preserving techniques. Fig. 6 and Table III illustrate the accuracy of each algorithm under varying privacy budgets ϵ. These baseline methods are introduced below:

- *DPSGDGCN and DS-DPSGDGCN:* Following the work of [45] and [7], we re-implement a differentially private stochastic gradient descent (DPSGD)-based GCN, namely DPSGDGCN. This method introduces DP in the back propagation learning procedure, which provides a strong privacy guarantee. In addition, to verify the validity of the degree sensitivity-based privacy-enhancement measure proposed in this paper, we constructed a degree sensitive-based DPSGDGCN, namely DS-DPSGDGCN.
- *LPGCN and DS-LPGCN:*The LPGCN was originally designed by [34], which introduces a multi-bit-based

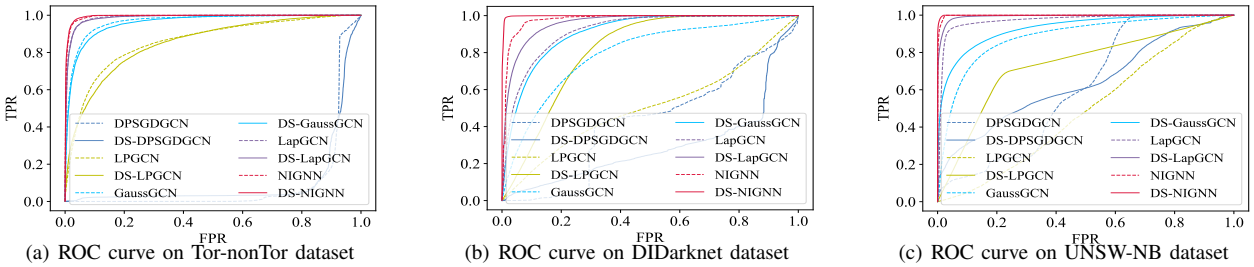
TABLE III: NIGCN versus private-preserving methods.

Methods	Tor-nontor Dataset			DIDraknet Dataset			UNSW-NB Dataset		
	ϵ=0.01	ϵ=1	ϵ=10	ϵ=0.01	ϵ=1	ϵ=10	ϵ=0.01	ϵ=1	ϵ=10
DPSGDGCN	11.94	38.87	40.79	17.62	22.56	81.45	20.34	40.03	51.89
LPGNN	88.87	88.88	88.21	76.59	79.38	80.55	77.14	77.20	79.66
GaussGCN	87.30	87.65	95.86	78.05	77.72	86.55	72.83	79.63	94.52
LapGCN	87.17	89.84	96.41	79.02	80.35	90.40	79.58	77.69	96.85
NIGNN	96.43	97.63	97.57	89.76	94.99	95.13	97.58	98.03	98.05
DS-DPSGDGCN	11.93	78.97	87.54	17.62	52.92	75.67	26.05	52.34	53.01
DS-LPGNN	89.66	89.84	89.45	77.63	79.95	79.00	78.09	77.32	79.66
DS-GaussGCN	86.81	88.37	95.72	80.09	81.11	89.22	76.82	79.27	95.84
DS-LapGCN	87.57	89.82	96.58	79.24	83.09	93.37	76.66	84.90	98.05
DS-NIGNN	97.09	97.69	97.76	91.96	98.55	98.81	98.60	98.88	98.91



(a) Comparisons on Tor-nonTor dataset (b) Comparisons on DIDarknet dataset (c) Comparisons on UNSW-NB dataset

Fig. 6: Experimental results of different private-preserving method comparisons under varying privacy budgets ϵ.



(a) ROC curve on Tor-nonTor dataset (b) ROC curve on DIDarknet dataset (c) ROC curve on UNSW-NB dataset

Fig. 7: ROC curve of NIGNN versus private-preserving methods on three benchmark datasets.

privacy-preserving mechanism to protect the node data privacy. Similar to DS-DPSGDGCN, we also built a degree-sensitive LPGCN, namely DS-LPGCN.

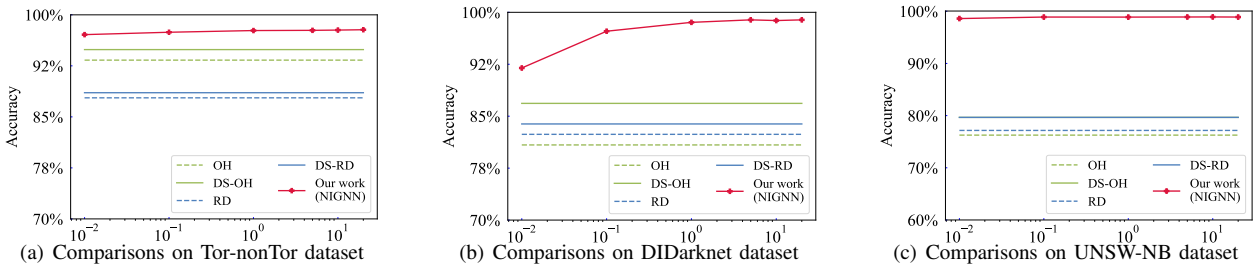
- *GaussGCN and DS-GaussGCN:* The GaussGCN uses the Gaussian-based DP mechanism in the input layer of the standard GCN, and injects the Gaussian noise [46] [37] [6] into the model inputs.
- *LapGCN and DS-LapGCN:* Similar to GaussGCN, the LapGCN injects the Laplace noise [46] [37] [6] sampled from the Laplacian distribution into the model inputs.
- *NIGCN and DS-NIGCN:* The two models use the CCP-based DP mechanism proposed in this paper to perturb the model inputs.

From the results, we can find that GaussGCN and LapGCN outperform DPSGDGCN on all benchmark datasets. This is because the DPSGDGCN can only be trained in a limited number of epochs. As the number of iterations of the model increases, the privacy budget ϵ noise and accumulate, reducing utility. Moreover, the baseline models (such as GaussGCN and LapGCN) incur significant errors on the three datasets when the ϵ is small (e.g., $\epsilon = 0.01$). The reason is that too much noise is added, making it difficult for the baseline models to converge to global optimization. The performance of the baseline models improves as the privacy budget ϵ increases. Importantly, we observe that NIGNN outperforms all other private-preserving methods in all cases. This proves the effectiveness of our approach. As shown in Fig. 6 and Table III, the performance of NIGNN shows almost no variations to the changes in ϵ on all benchmark dataset. This means that NIGNN allows using smaller values of ϵ for better privacy protection without sacrificing much of its accuracy. Fig. 7 provides the ROC results of all baselines on the three datasets. In summary, NIGNN can achieve the privacy-utility trade-off. Specifically, we also found that classification performance is further improved when the degree-sensitive mechanism is employed. This is because the degree sensitivity mechanism can mitigate the effect of noise.

C. Robustness to Attribute Inference Attacks

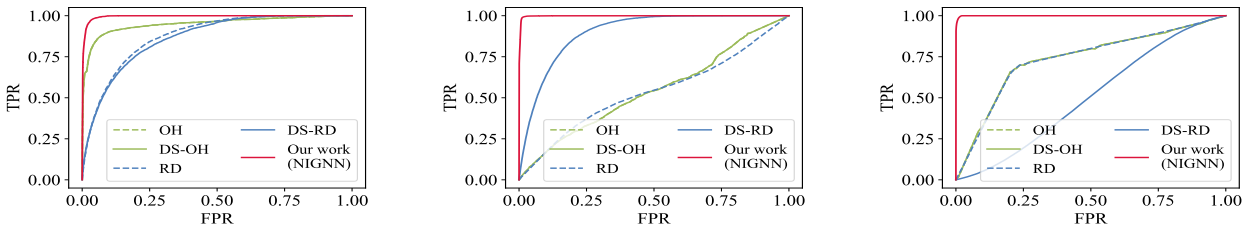
In this experiment, we investigate the accuracy of attribute inference attacks under both non-private and privacy-preserving NIGNN frameworks. As illustrated in Table IV, the Rand attack approach serves as the baseline method for predicting users' sensitive attributes. It randomly assigns predictions to sensitive attributes without leveraging any information from the dataset. As expected, the Rand

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3417853



(a) Comparisons on Tor-nonTor dataset (b) Comparisons on DIDarknet dataset (c) Comparisons on UNSW-NB dataset

Fig. 8: Experimental results of different feature comparisons under varying privacy budgets ϵ.



(a) ROC curve on Tor-nonTor dataset (b) ROC curve on DIDarknet dataset (c) ROC curve on UNSW-NB dataset

Fig. 9: Experimental results of different feature comparisons.

attack approach achieved close to 50% accuracy on all three datasets, which is consistent with random guess expectations. Following previous work [35], we re-implemented the MB attack model to infer sensitive attributes. In the case of the MB attack, we assume that the attacker possessed partial knowledge of the training dataset. We then evaluated the performance of the MB attack model under both non-private and privacy-preserving settings. In the non-private setting, the MB attack model achieved high accuracies of 85.73%, 98.91%, and 97.36% on the three datasets, respectively. These results show that MB attack models can effectively and accurately infer sensitive attributes in the absence of privacy protection mechanisms. However, under the privacy-preserving setting, the accuracy of the MB-AT attack model is significantly reduced. Additionally, following previous work [47], we re-implemented the RI-MI and FP-MA methods, which were originally developed to reconstruct missing attributes. The results summarized in Table IV indicate that RI-MI performs relatively poorly, achieving similar performance to the Rand attack method. In contrast, FP-MA achieves better attack accuracy in the non-privacy setting. However, the proposed privacy-preserving NIGNN significantly reduces the attack accuracy of both RI-MI and FP-MA. For instance, the attack accuracy of FP-MA is reduced by 33.7%, 69.58%, and 79.7% on the three datasets, respectively. These results demonstrate the effectiveness of the privacy-preserving NIGCN in mitigating attribute inference attacks. In conclusion, the accuracy of attribute inference attacks differs significantly between non-private and privacy-preserving NIGNN frameworks. While non-private NIGNN is vulnerable to attribute inference attacks, the privacy-preserving NIGNN offers a more robust defense against such attacks by leveraging techniques such as differential privacy.

TABLE IV: Accuracy of attribute inference attack under non-private and privacy-preserving NIGNN.

Methods		Tor-nontor	DIDraknet	UNSW-NB
Rand	non-private	50.42	50.24	49.77
	privacy-preserving	50.42	50.24	49.77
MB	non-private	85.73	92.56	97.36
	privacy-preserving	55.49	64.71	88.46
RI-MI	non-private	50.20	45.39	45.94
	privacy-preserving	44.81	14.15	11.95
FA-MA	non-private	67.16	84.81	87.55
	privacy-preserving	33.46	15.23	7.85

TABLE V: Effect of different features.

Datasets	OH	DS-OH	RD	DS-RD	Our work	
Tor-nonTor	93.34	94.89	87.80	88.55	ϵ = 0.01	97.09
					ϵ = 1	97.76
					ϵ =10	97.76
DIDarknet	80.85	86.85	82.38	83.88	ϵ = 0.01	91.96
					ϵ = 1	98.55
					ϵ =10	98.81
UNSW-NB	76.26	79.66	77.16	79.66	ϵ = 0.01	98.60
					ϵ = 1	98.88
					ϵ =10	98.91

TABLE VI: Effect of different graph-less NIDS methods.

Methods	Tor-nonTor	DIDarknet	UNSW-NB
LR	93.52	94.02	97.73
SVM	94.31	96.91	98.49
DNN	94.91	98.24	98.89
CNN	94.90	97.96	98.75
RNN	95.44	98.04	98.85
LSTM	95.12	98.52	98.63
NIGNN	97.66	98.83	98.92

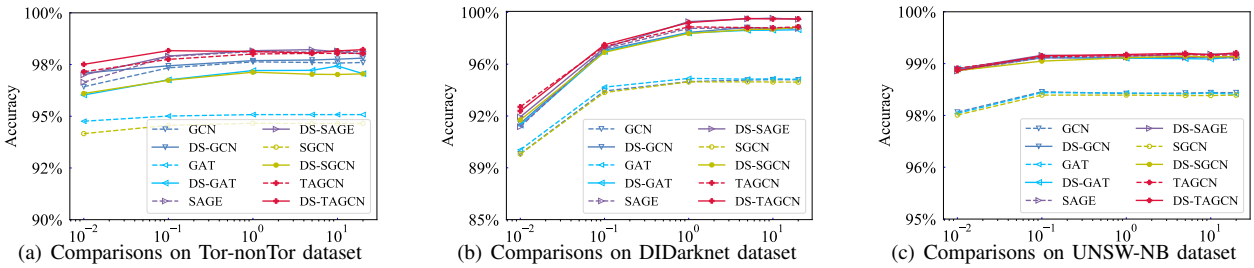
D. The Influence of Different Features

To show the effectiveness of the traffic-based vertex features used in this paper, we compared the classification performance of our feature extraction with some other methods.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3417853

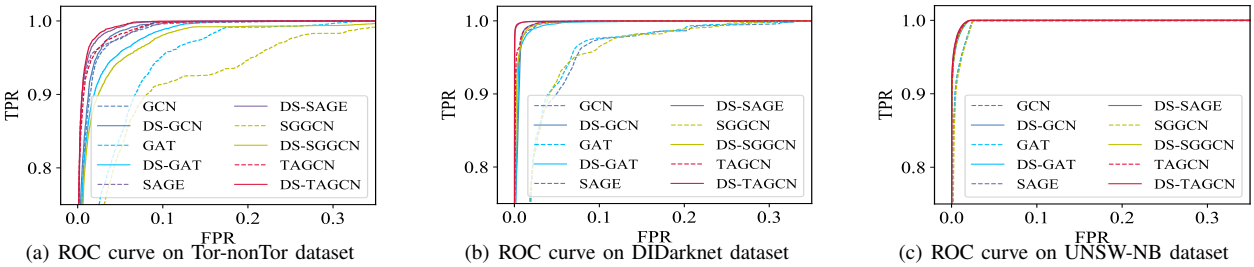
TABLE VII: NIGNN versus graph-based models.

Models	Tor-nontor Dataset				DIDraknet Dataset				UNSW-NB Dataset			
	ϵ=0.01	ϵ=1	ϵ=10	noDP	ϵ=0.01	ϵ=1	ϵ=10	noDP	ϵ=0.01	ϵ=1	ϵ=10	noDP
GCN	96.43	97.63	97.57	97.66	89.76	94.99	95.13	98.83	97.58	98.03	98.05	98.92
GAT	94.76	95.08	95.08	95.12	90.05	95.22	95.19	95.27	97.55	98.04	98.04	98.03
SAGE	96.65	98.13	98.16	98.13	91.72	98.84	98.84	98.82	98.57	98.91	98.90	98.89
SGCN	94.16	94.66	94.65	94.65	89.75	94.97	94.96	94.95	97.51	97.99	97.98	97.99
TAGCN	97.15	98.01	98.04	98.04	93.16	98.95	98.88	98.92	98.64	98.95	98.95	98.97
DS-GCN	97.09	97.69	97.76	97.66	91.96	98.55	98.81	98.83	98.60	98.88	98.91	98.92
DS-GAT	96.03	97.21	97.43	95.12	91.83	98.51	98.71	95.27	98.62	98.88	98.86	98.03
DS-SAGE	97.00	98.17	98.12	98.13	92.43	99.33	99.58	98.82	98.62	98.93	98.98	98.89
DS-SGCN	96.10	97.13	97.02	94.65	92.21	98.47	98.78	94.95	98.58	98.89	98.91	97.99
DS-TAGCN	97.51	98.13	98.16	98.04	92.86	99.26	99.52	98.92	98.58	98.97	98.96	98.97



(a) Comparisons on Tor-nonTor dataset (b) Comparisons on DIDarknet dataset (c) Comparisons on UNSW-NB dataset

Fig. 10: Experimental results of different graph-based model comparisons under varying privacy budgets ϵ.



(a) ROC curve on Tor-nonTor dataset (b) ROC curve on DIDarknet dataset (c) ROC curve on UNSW-NB dataset

Fig. 11: ROC curve of NIGNN versus graph-based models on three benchmark datasets.

Fig. 8 and Table V present the evaluation results, providing valuable insights into the performance of different models. These baseline methods are introduced below:

- *OH and DS-OH:* Following the work of [34] and [9], the OH method uses the one-hot encoding of vertex degrees. A standard model of GCN originally proposed by [10] was then used for training. Specifically, we set the feature dimension equal to the originally provided traffic-based vertex features.
- *RD and RD-OH:* The RD [34] consists of a four-layer GCN, which randomly initializes the input features with a Gaussian distribution. The two learning models are optimized with stochastic gradient descent (SGD).

Specifically, these methods can be regarded as "fully private" because they do not require any private vertex features. In Fig. 9, we see that OHTGCN and RANGCN yield inferior performance. The OHTGCN model gives similar performances to the RANGCN model on the three datasets. Our method performs considerably better than these two fully private methods, i.e., OHTGCN and RANGCN. Experimental results demonstrate the effectiveness of traffic-based vertex features used in this paper. This means that vertex features are helpful for network intrusion detection.

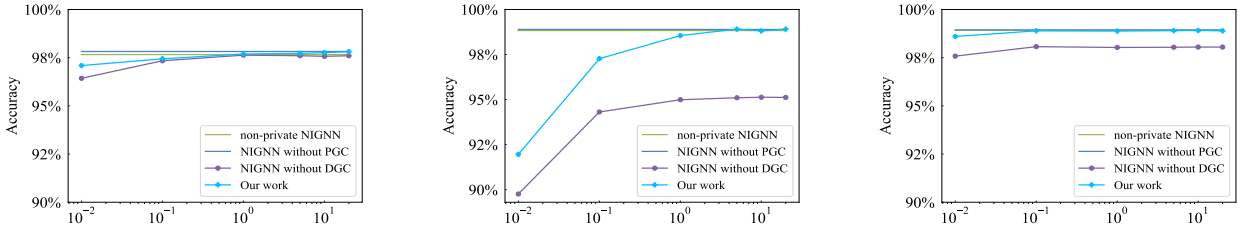
TABLE VIII: Effect of different components.

Datasets	non-private NIGNN	NIGNN without PGC	NIGNN without DGC		Our work
Tor-nonTor	97.66	97.82	ϵ = 0.01	96.43	97.09
			ϵ = 1	97.63	97.69
			ϵ = 10	97.57	97.76
DIDarknet	98.83	98.89	ϵ = 0.01	89.76	91.96
			ϵ = 1	94.99	98.55
			ϵ = 10	95.13	98.81
UNSW-NB	98.92	98.94	ϵ = 0.01	97.58	98.60
			ϵ = 1	98.03	98.88
			ϵ = 10	98.05	98.91

E. Comparison with Graph-Less NIDS Methods

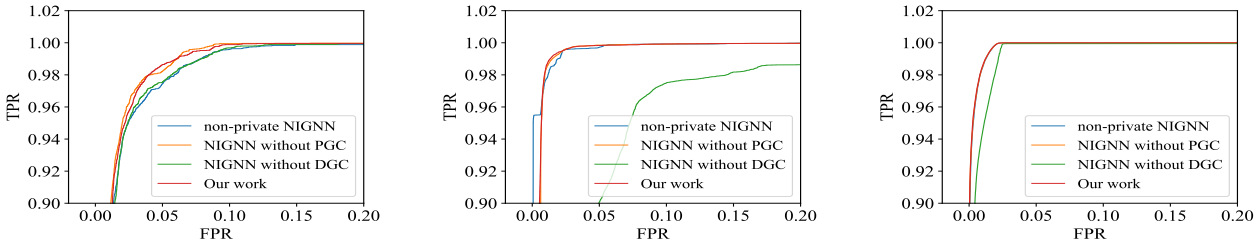
In this experiment, we compared the proposed graph-based NIGNN with several state-of-the-art graph-less NIDS methods. Table VI shows the results. Notably, linear regression (LR) and support vector machine (SVM) perform the worst among the graph-less NIDS methods. This suggests that they may not be able to capture the complex and nonlinear patterns in network traffic data, limiting their ability to accurately detect network intrusions. In contrast, neural network models including deep neural networks (DNN), convolutional neural networks (CNN), recurrent neural networks (RNN), and

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3417853



(a) Comparisons on Tor-nonTor dataset (b) Comparisons on DIDarknet dataset (c) Comparisons on UNSW-NB dataset

Fig. 12: Effect of different components, wrt. privacy budget ϵ, with DP.



(a) ROC curve on Tor-nonTor dataset (b) ROC curve on DIDarknet dataset (c) ROC curve on UNSW-NB dataset

Fig. 13: Ablation studies of NIGNN on three benchmark datasets.

long short-term memory (LSTM) networks can better fit the data distribution, resulting in higher detection performance. However, it is noteworthy that even though these neural network models outperform LR and SVM in graph-less NIDS methods, they are still not comparable to the proposed graph-based NIGNN method. This result demonstrates the superiority of the proposed NIGNN in detecting network intrusions, and emphasizes the unique advantages of graph-based approaches in modeling the complex graph structures and relationships within network traffic data. By leveraging the graph structure of network data, NIGNN can capture complex interactions between network entities and effectively detect network intrusions.

F. Comparison with Different Graph Neural Networks

In our experiment, we included some state-of-the-art GNN models, i.e., GCN [10], GAT [25], SAGE [26], SGCN [48], and TAGCN [49], which are commonly used for graph-related tasks. To highlight the significance of the proposed NIGNN framework, we re-implemented these baseline models in the NIGNN framework. Fig. 10 and Table VII show the accuracy results of each model across varying privacy budgets. From the results, we find that the baseline GCN performs considerably better than GAT and SGCN but are almost worse than the SAGE. The accuracy gain is increased by roughly 3.5% on the three datasets. It indicates that SAGE can capture useful information. Specifically, TAGCN performs better than the baseline model SAGE. This is because TAGCN utilizes a set of fixed-size filters to perform graph convolution without the need for convolution approximation, enabling efficient extraction of local features. Therefore, we recommend adopting the TAGCN model in the NIGNN framework. When ϵ is set to 1, the accuracy loss is less than 1% in the worst case, demonstrating the effectiveness of our privacy-preserving method. This is because the aggregate function in the graph convolutional layer can eliminate most of the noise in the vertex features. The results clearly show that the proposed NIGNN not only preserves the power of the original NIGNN on the detection task but also provides compelling evidence on improving privacy and utility.

Directly injecting noise to the model input would incur much noise due to the aggregation operation. To alleviate this problem, this paper proposes a graph convolution layer based on degree sensitivity privacy-enhancement mechanism, which manipulates the outputs of this layer by giving vertices with very few neighbors a relatively high influence on model learning. To highlight the significance of the proposed degree sensitivity-based privacy-enhancement mechanism, we introduce the degree sensitivity privacy-enhancement mechanism into these baseline models, i.e., DS-GCN, DS-GAT, DS-SAGE, DS-SGCN, and DS-TAGCN. Fig. 11 depicts the ROC results of the performance comparison. We can see that using degree sensitivity mechanism can effectively improve model accuracy. This verifies the validity of the proposed degree sensitivity privacy-enhancement mechanism.

G. Ablation experiment

To demonstrate the contribution of each module in the NIGNN framework, we conducted a set of experiments. To have a fair comparison, the non-private NIGNN adopts a standard three-layer GCN as a baseline model. Compared to these baseline models, the performance of the proposed private-preserving NIGNN is closest to that of the non-private NIGNN on all three benchmark datasets. Fig. 12, Fig. 13 and Table VIII show the results. We observe that the accuracy is almost identical to that of non-private NIGNN, demonstrating its strong generalization ability. Our model can achieve high detection accuracy while effectively protecting user privacy.

Futhermore, we test the effects of the PGC and DGC on performance, respectively, i.e. NIGNN without PGC (with DGC) and NIGNN without DGC (with PGC). We can see that without the PGC layer, the baseline model (NIGNN without PGC) is comparable to the non-private NIGNN and even exceeds it on the Tor-nonTor and UNSW-NB datasets. This is because this model only uses the DGC layer, which shows that the DGC layer is useful and can effectively improve model performance. Moreover, it can be observed that the baseline model (NIGNN without DGC) has the worst performance on all datasets. This is not surprising, as the PGC layer used in this model introduces a lot of DP noise into the model input, which results in degraded model performance.

VII. Conclusion

The existing deep learning-based methods do not consider the topologic structure information of network traffic graph. This paper proposes a graph construction method to transform network traffic data into graph structures. Then, the network intrusion detection can be converted to a specific type of node classification task. Moreover, we built a differential privacy-based graph representation learning model that is trained within an appropriate privacy budget ϵ. In fact, our privacy-preserving method can be easily generalized to other tasks. Moreover, we design a degree-sensitive privacy-enhancement measure to exert different effects on vertices, which can reduce the noise introduced by the DP mechanism while effectively learning an accurate model. We evaluated the performance of NIGNN on three datasets for network intrusion detection. Experimental results demonstrate that NIGNN can achieve a high detection rate close to the non-privacy ones, while providing a rigorous privacy guarantee. The superior performance of NIGNN highlights that privacy-preserving GNN is a worthwhile exploration.

VIII. Acknowledgement

This work was supported by the National Natural Science Foundation of China Project (62172441, 62172449), the Joint Funds for Railway Fundamental Research of National Natural Science Foundation of China (U2368201), special fund of National Key Laboratory of Ni&Co Associated Minerals Resources Development and Comprehensive Utilization (GZSYS-KY-2022-018, GZSYS-KY-2022-024), Key Project of Shenzhen City Special Fund for Fundamental Research (JCYJ20220818103200002), and the National Natural Science Foundation of Hunan Province (2023JJ30696).

References

[1] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.

[2] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[3] J. Zheng and D. Li, "Gcn-tc: Combining trace graph with statistical features for network traffic classification," in *Proceedings of ICC*, 2019, pp. 1–6.

[4] Y. Zhu, J. Tao, H. Wang, L. Yu, Y. Luo, T. Qi, Z. Wang, and Y. Xu, "Dgnn: Accurate darknet application classification adopting attention graph neural network," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2023.

[5] K. Pan, M. Gong, K. Feng, and K. Wang, "Differentially private regression analysis with dynamic privacy allocation," *Knowl. Based Syst.*, vol. 217, p. 106795, 2021.

[6] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of CCS*, 2016, pp. 308–318.

[7] J. Lee and D. Kifer, "Concentrated differentially private gradient descent with adaptive per-iteration privacy budget," in *Proceedings of ACM SIGKDD*, 2018, pp. 1656–1665.

[8] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *Proceedings of ICLR*, 2018.

[9] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks," in *Proceedings of ICLR*, 2018.

[10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of ICLR*, 2016.

[11] D. E. Kim and M. Gofman, "Comparison of shallow and deep neural networks for network intrusion detection," in *Proceedings of CCWC*, 2018, pp. 204–208.

[12] J. Liu, C. Zhang, and Y. Fang, "Epic: A differential privacy framework to defend smart homes against internet traffic analysis," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1206–1217, 2018.

[13] S. Das, M. Ashrafuzzaman, F. T. Sheldon, and S. Shiva, "Network intrusion detection using natural language processing and ensemble machine learning," in *Proceedings of SSCI*, 2020, pp. 829–835.

[14] J. Liu, C. Zhang, B. Lorenzo, and Y. Fang, "Dpavatar: A real-time location protection framework for incumbent users in cognitive radio networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 3, pp. 552–565, 2020.

[15] Y.-F. Hsu and M. Matsuoka, "A deep reinforcement learning approach for anomaly network intrusion detection system," in *Proceedings of CloudNet*, 2020, pp. 1–6.

[16] T. Elahi, G. Danezis, and I. Goldberg, "Privex: Private collection of traffic statistics for anonymous communication networks," in *Proceedings of CCS*, 2014, pp. 1068–1079.

[17] M. Keshk, E. Sitnikova, N. Moustafa, J. Hu, and I. Khalil, "An integrated framework for privacy-preserving based anomaly detection for cyber-physical systems," *IEEE Transactions on Sustainable Computing*, vol. 6, no. 1, pp. 66–79, 2021.

[18] L. Sgaglione, L. Coppolino, S. D'Antonio, G. Mazzeo, L. Romano, D. Cotroneo, and A. Scognamiglio, "Privacy preserving intrusion detection via homomorphic encryption," in *International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises*. IEEE, 2019, pp. 321–326.

[19] L. Mokry, P. Slife, P. Bishop, J. Quiroz, C. Guzzi, Z. Chen, A. Crainiceanu, and D. Needham, "Efficient and privacy-preserving collaborative intrusion detection using additive secret sharing and differential privacy," in *2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, December 15-18, 2021*. IEEE, 2021, pp. 3324–3333.

[20] A. Alazab, A. Khraisat, S. Singh, and T. Jan, "Enhancing privacy-preserving intrusion detection through federated learning," *Electronics*, vol. 12, no. 16, p. 3382, 2023.

[21] D. Jin, S. Chen, H. He, X. Jiang, S. Cheng, and J. Yang, "Federated incremental learning based evolvable intrusion detection system for zero-day attacks," *IEEE Netw.*, vol. 37, no. 1, pp. 125–132, 2023.

[22] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Skeleton-based action recognition with directed graph neural networks," in *Proceedings of CVPR*, 2019, pp. 7912–7921.

[23] Z. Li, Z. Cui, S. Wu, X. Zhang, and L. Wang, "Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction," in *Proceedings of ICIKM*, 2019, pp. 539–548.

[24] J. Kim, T. Kim, S. Kim, and C. D. Yoo, "Edge-labeling graph neural network for few-shot learning," in *Proceedings of CVPR*, 2019, pp. 11–20.

[25] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proceedings of ICLR*, 2018.

[26] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of NIPS*, vol. 30, 2017, pp. 1024–1034.

[27] H. Zhang, B. Wu, X. Yuan, S. Pan, H. Tong, and J. Pei, "Trustworthy graph neural networks: Aspects, methods and trends," *CoRR*, vol.

abs/2205.07424, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2205.07424

[28] Y. Zhang, Y. Zhao, Z. Li, X. Cheng, Y. Wang, O. Kotevska, P. S. Yu, and T. Derr, "A survey on privacy in graph neural networks: Attacks, preservation, and applications," *CoRR*, vol. abs/2308.16375, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2308.16375

[29] S. Wang, Y. Zheng, and X. Jia, "Secgnn: Privacy-preserving graph neural network training and inference as a cloud service," *IEEE Trans. Serv. Comput.*, vol. 16, no. 4, pp. 2923–2938, 2023.

[30] C. Wu, F. Wu, Y. Cao, Y. Huang, and X. Xie, "Fedgnn: Federated graph neural network for privacy-preserving recommendation," *CoRR*, vol. abs/2102.04925, 2021.

[31] X. Miao, W. Zhang, Y. Jiang, F. Fu, Y. Shao, L. Chen, Y. Tao, G. Cao, and B. Cui, "P^2cg: a privacy preserving collaborative graph neural network training framework," *VLDB J.*, vol. 32, no. 4, pp. 717–736, 2023.

[32] B. Zhang, M. Luo, S. Feng, Z. Liu, J. Zhou, and Q. Zheng, "PPS-GCN: A privacy-preserving subgraph sampling based distributed GCN training method," *CoRR*, vol. abs/2110.12906, 2021.

[33] K. Bhaila, W. Huang, Y. Wu, and X. Wu, "Local differential privacy in graph neural networks: a reconstruction approach," *CoRR*, vol. abs/2309.08569, 2023.

[34] S. Sajadmanesh and D. Gatica-Perez, "Locally private graph neural networks," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2021, pp. 2130–2145.

[35] W. Du, X. Ma, W. Dong, D. Zhang, C. Zhang, and Q. Sun, "Calibrating privacy budgets for locally private graph neural networks," in *2021 International Conference on Networking and Network Applications, NaNA 2021, Lijiang City, China, October 29 - Nov. 1, 2021*. IEEE, 2021, pp. 23–29. [Online]. Available: https://doi.org/10.1109/NaNA53684.2021.00012

[36] W. Lin, B. Li, and C. Wang, "Towards private learning on decentralized graphs with local differential privacy," *IEEE Trans. Inf. Forensics Secur.*, vol. 17, pp. 2936–2946, 2022.

[37] M. U. Hassan, M. H. Rehmani, and J. Chen, "Differential privacy techniques for cyber physical systems: A survey," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 1, pp. 746–789, 2020.

[38] N. Li, M. Lyu, D. Su, and W. Yang, *Differential Privacy: From Theory to Practice*, ser. Synthesis Lectures on Information Security, Privacy, & Trust. Morgan & Claypool Publishers, 2016.

[39] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 3–18.

[40] K. H. Zou, K. Tuncali, and S. G. Silverman, "Correlation and simple linear regression," *Radiology*, vol. 227, no. 3, pp. 617–622, 2003.

[41] J. C. F. de Winter, S. D. Gosling, and J. Potter, "Comparing the pearson and spearman correlation coefficients across distributions and sample sizes: A tutorial using simulations and empirical data." *Psychological Methods*, vol. 21, no. 3, pp. 273–290, 2016.

[42] A. H. Lashkari, G. Kaur, and A. Rahali, "Didarknet: A contemporary approach to detect and characterize the darknet traffic using deep image learning," in *Proceedings of ICCNS*, 2020.

[43] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features." in *Proceedings of ICISSP*, 2017, pp. 253–262.

[44] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *Proceedings of MilCIS*, 2015, pp. 1–6.

[45] T. T. Mueller, J. C. Paetzold, C. Prabhakar, D. Usynin, D. Rueckert, and G. Kaissis, "Differentially private graph neural networks for whole-graph classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 6, pp. 7308–7318, 2023.

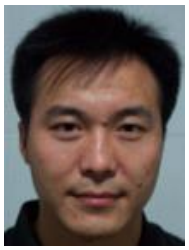
[46] L. Zhao, Q. Wang, Q. Zou, Y. Zhang, and Y. Chen, "Privacy-preserving collaborative deep learning with unreliable participants," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1486–1500, 2020.

[47] I. E. Olatunji, A. Hizber, O. Sihlovec, and M. Khosla, "Does black-box attribute inference attacks on graph neural networks constitute privacy risk?" *CoRR*, vol. abs/2306.00578, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2306.00578

[48] F. Wu, A. H. S. Jr., T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proceedings of ICML*, vol. 97, 2019, pp. 6861–6871.

[49] J. Du, S. Zhang, G. Wu, J. M. F. Moura, and S. Kar, "Topology adaptive graph convolutional networks," *CoRR*, vol. abs/1710.10370, 2017.

Xinjun Pei is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Central South University, China. Since 2017, he has been engaged in the direction of information security. His research interests include deep learning, edge computing and IoT security.



Xiaoheng Deng (Senior Member, IEEE) received the Ph.D. degree in computer science from Central South University, Changsha, Hunan, P.R. China, in 2005. He is currently a Full Professor, Dean of School of Electronic Information, Central South University. He is a Joint researcher of Shenzhen Research Institute, Central South University. He is a senior member of IEEE, a senior member of CCF, a member of CCF Pervasive Computing Council, and a member of ACM. He has been a chair of CCF YOCSEF CHANGSHA from 2009 to 2010. His research interests include network security, edge computing, Internet of Things, online social network analysis, data mining, and pattern recognization.

Shengwei Tian received the B.S., M.S., and Ph.D. degrees from the School of Information Science and Engineering, Xinjiang University, China, in 1997, 2004 and 2010, respectively. He is currently a Full Professor, Dean of School of Software, Xinjiang University. His research interests include artificial intelligence, natural language processing, and cyberspace security.

Ping Jiang received the master's degree in computer engineering from the University of Western Ontario, Canada, in 2018. He is currently pursuing a Ph.D. degree with the Electrical and Communication Engineering Department, Central South University, China. His research interests include machine learning, natural language processing, computer vision, and security issues in neural networks.

Yunlong Zhao received the bachelor's degree from the School of Computer Science, Beijing University of Posts and Telecommunications, in 2020. He is currently pursuing a Ph.D. degree withthe school of Computer Science and Engineering, Central South University, China. Since 2021, he has been engaged in the direction of privacy protection. His research interests include federated learning, edge computing, and security issues in neural networks.

Kaiping Xue (M'09-SM'15) received his bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and received the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. From May 2012 to May 2013, he was a postdoctoral researcher with the Department of Electrical and Computer Engineering, University of Florida. Currently, he is a Professor in the School of Cyber Security, USTC. His research interests include next-generation Internet architecture design, transmission optimization and network security. He serves on the Editorial Board of several journals, including the IEEE Transactions on Dependable and Secure Computing (TDSC), the IEEE Transactions on Wireless Communications (TWC), and the IEEE Transactions on Network and Service Management (TNSM). He has also served as a (Lead) Guest Editor for many reputed journals/magazines, including IEEE Journal on Selected Areas in Communications (JSAC), IEEE Communications Magazine, and IEEE Network. He is an IET Fellow and an IEEE Senior Member.