Privacy-Enhanced Graph Neural Network for Decentralized Local Graphs

Xinjun Pei, *Student Member, IEEE*, Xiaoheng Deng, *Senior Member, IEEE*, Shengwei Tian, Jianqing Liu, *Member, IEEE*, and Kaiping Xue, *Senior Member, IEEE*

***Abstract*— With the ever-growing interest in modeling complex graph structures, graph neural networks (GNN) provide a generalized form of exploiting non-Euclidean space data. However, the global graph may be distributed across multiple data centers, which makes conventional graph-based models incapable of modeling a complete graph structure. This also brings an unprecedented challenge to user privacy protection in distributed graph learning. Due to privacy requirements of legal policies, existing graph-based solutions are difficult to deploy in practice. In this paper, we propose a privacy-preserving graph neural network based on local graph augmentation, named LGA-PGNN, which preserves user privacy by enforcing local differential privacy (LDP) noise into the decentralized local graphs held by different data holders. Moreover, we perform local neighborhood augmentation on low-degree vertices to enhance the expressiveness of the learned model. Specifically, we propose two graph privacy attacks, namely attribute inference attack and link stealing attack, which aim at compromising user privacy. The experimental results demonstrate that LGA-PGNN can effectively mitigate these two attacks and provably avoid potential privacy leakage while ensuring the utility of the learning model.**

***Index Terms*— Local differential privacy, graph augmentation, graph convolutional network, privacy-preserving.**

I. Introduction

THE large amount of graph data generated from the Internet of Things (IoT) devices underscores the demand for advanced graph analysis techniques in order to model complex graph structures. The emergence of graph neural networks (GNN) have attracted increasing attentions from academia and industry perspectives, since it allows to directly model the high-dimensional vertex features and high-order interactions between vertices. In many real-world scenarios, graph-structured data is usually sensitive and may contain user private information, such as a user's friend list, profile information, likes and comments in a social network [1], [2], [3], [4]. While anonymity on the Internet separates one's network identity from one's online activities [5], any collection and sharing of graph data can compromise the anonymity of users, thereby putting their privacy at risk. The proliferation of high-quality graph structured data has brought unprecedented challenges for user privacy. How to protect data privacy from being leaked during data sharing still remains an open problem.

The highly privacy-sensitive graphs in the real world make most of conventional GNN models infeasible because they cannot defend against privacy inference attacks on graph data. We cannot exclude the possibility that a strong adversary can invert the sensitive information through techniques such as membership inference attacks [6], [7], [8], model inversion attacks [9], and model extraction attacks [10]. This privacy risk is further exacerbated by the increasing trend of publishing and sharing pre-trained models. In our problem setting, we propose two graph privacy attacks to infer privacy vertex attributes or links of a user's local graph.

To mitigate privacy threats, local differential privacy (LDP) has become an indispensable and well-recognized defense method. Unlike traditional centralized differential privacy (CDP), which typically assumes that a trusted data curator can see the true data [11], [12], LDP perturbs each user's data records locally before sending them to the curator. The curator then performs calculations on the collected perturbed data records to estimate statistical analysis results on the original data. As a result, LDP can provide users with a stronger privacy guarantee than CDP. Some recent works [13], [14] have combined LDP with various deep learning algorithms to alleviate privacy issues persisted in these algorithms. However, few research works have considered guarding against privacy inference attacks on graph data. This emphasizes the need for privacy-preserving GNN techniques. In this paper, we focus on understanding the risks of vertex and edge privacy when training and releasing a GNN model, ensuring that the released model and its predictions do not reveal private information about the undisclosed training data.

Manuscript received 24 May 2023; revised 27 August 2023; accepted 23 September 2023. Date of publication 2 November 2023; date of current version 19 December 2023. This work was supported in part by the National Natural Science Foundation of China Project under Grant 62172441 and Grant 62172449; in part by the Joint Funds for Railway Fundamental Research of National Natural Science Foundation of China under Grant U2368201; in part by the Special Fund of National Key Laboratory of Ni&Co Associated Minerals Resources Development and Comprehensive Utilization under Grant GZSYS-KY-2022-018 and Grant GZSYS-KY-2022-024; in part by the Key Project of Shenzhen City Special Fund for Fundamental Research under Grant JCYJ20220818103200002; in part by the National Natural Science Foundation of Hunan Province under Grant 2023JJ30696; and in part by the Autonomous Region Key Research and Development Project under Grant 2021B01002. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Grigorios Loukides. *(Corresponding author: Xiaoheng Deng.)*

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

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

Jianqing Liu is with the Department of Computer Science, North Carolina State University, Raleigh, NC 27606 USA (e-mail: jliu96@ncsu.edu).

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).

This article has supplementary downloadable material available at https://doi.org/10.1109/TIFS.2023.3329971, provided by the authors.

Digital Object Identifier 10.1109/TIFS.2023.3329971

Most existing studies on GNN assume unrestricted access to graph data, ignoring potential privacy risks in data sharing, especially in graph isolation scenarios where the graph data is generated locally on each user device and remains decentralized [1], [2], [4], [5], [15]. For example, a social graph with users' purchase history held by a multinational corporation may be distributed across multiple data centers in different countries [16]. The server collects user's behavioral data (i.e. vertex features) and interaction data (i.e. neighborhood information) to enrich the GNN, and then provides users with a GNN inferrence API for queries. Moreover, in the case of a GNN, each neighboring vertex pair may exchange their hidden vector representations with each other multiple times through a message-passing mechanism [17], [18], [19], [20], [21], [22]. In a graph isolation scenario, aggregating sufficient local neighborhood information for low-degree vertices with few neighbors is highly challenging, and requires thoughtful design considerations. To cope with this problem, existing methods [4] incorporate multi-hop neighboring information by expanding the receptive field, but may lead to over-smoothing. A promising solution is to develop a graph augmentation strategy to enhance the local neighborhood representation for each vertex, which enhances the expressiveness of the GNN model.

To cope with the graph isolation problem, we propose a privacy-preserving graph convolutional network (LGA-PGNN) to provide privacy protection for user's data records. To prevent adversaries from inferring sensitive information, we enforce LDP on decentralized local graphs held by different users and then train LGA-PGNN with a privacy graph convolutional layer. Moreover, we develop a local graph augmentation (LGA) method that expands the local neighborhoods of low-degree vertices by generating vertex augmentations, which enhances the expressiveness of the learned model. We present two types of graph privacy attacks (i.e., attribute inference attack and link stealing attack) and how they steal users' private information. We demonstrate that LGA-PGNN can significantly mitigate these two attacks and reduce the attack performance by 79.23% and 64.34%, respectively. Overall, the proposed LGA-PGNN can guarantee the data utility and allow an accurate model to be learned efficiently, while provably avoiding potential privacy leakages. To our knowledge, we are not aware of existing privacy-preserving GNNs for similar privacy attacks, as most previous locally private GNN models focused on defending against member inference attacks. Our major contributions are as follows:

- We propose LGA-PGNN, which enforces LDP in decentralized local graphs held by different data holders without consuming too much privacy budget, and utilizes a privacy graph convolutional layer to train the learning model. Specifically, we present two types of graph privacy attacks and how they steal users' private information.
- We develop LGA to expand the local neighborhood by generating vertex augmentations, which can act as a denoising mechanism. This enhances the expressiveness of the learned model.
- We conduct extensive experiments on real-world datasets to evaluate the effectiveness of LGA-PGNN. The experimental results show that LGA-PGNN can achieve high accuracy close to the LDP-exempt ones, and has the ability to maintain a sound privacy-accuracy trade-off.

The rest of the paper is organized as follows. we review the related work in Section II and give some preliminaries in Section III. Then, we describe the system model and threat model in Section IV. In Section V, we give the LGA-PGNN design details. Section VI analyzes the experimental results. Finally, Section VII concludes the paper.

II. Related Work

Recently, GNN has been considered as an emerging research area and has been successfully applied in many fields. The ability to model irregular graph data is the key to the success of GNNs, and numerous variants have been proposed, such as graph convolutional networks (GCN) [17], graph attention networks (GAT) [18], Chebyshev [19], SAGE [20], SimGCN [21], TAGCN [22] and so on. These methods recursively use neighborhood aggregation and transformation to compute the vertex representations. In this case, the graph structure is encoded into the neural network to improve classification performance. While the success of GNNs is undeniable, there are growing privacy concerns in the training data used to build these learning models. Several recent studies have attempted to reveal potential privacy attacks in the field of deep learning. For example, Shokri et al. [6] proposed a membership inference attack where the adversary can determine whether a data record belongs to the target model's training dataset through an inference model. In [23], the authors proposed a model inversion attack, where the adversary can access the learning model and extract memorized information from the model.

Several recent studies have investigated the feasibility of the attribute inference attacks and link stealing attacks. For attribute inference attack, traditional methods utilize the target user's friend information and community membership information to infer the target user's private attributes. For example, Gong and Liu [24] utilized publicly available social friends and user's behavioral records to infer attributes of the target user. Kosinski et al. [25] demonstrated that users' preference signals are highly vulnerable to attribute inference attacks. The most relevant work to us is [26], which presents a finer-grained concept of attribute inference in which an adversary can identify records with sensitive attributes from a candidate set with a high degree of confidence. They showed that trained models leak considerable information about the underlying training distribution that can be exploited to infer sensitive attributes about individuals. However, these methods do not discuss inference attacks on the graph's vertex attributes. Moreover, due to the complexity of graph data, these methods cannot be extended trivially to node-level privacy setting. In our problem setting, the server is not trusted and thus does not have direct access to the graph data. We assume that the adversary has partial knowledge of some training records and then use them to infer sensitive attribute values.

Little work focuses on protecting users against link stealing attacks. He et al. [27] proposed a black-box attack aimed at inferring sensitive links between nodes of interest. In this work, they discussed several types of background knowledge for attackers, including node attributes, partial graph information, and the utilization of a shadow dataset. Wu et al. [28] is the first work that proposed a link stealing attack framework, LINKTELLER, to infer the private edge information. In the inference process, LINKTELLER first queries the GNN inference API with a set of inference nodes, and then infers connections between nodes of interest using the returned prediction probability vectors. Similarly, Kolluri et al., [29] extended the LINKTEELER approach, and developed a privacy-preserving GNN model that is trained on graphs with privacy-sensitive edges. However, these methods have different problem settings that make them not directly and fairly comparable to our method. Such methods drastically change the adjacency matrix and severely distort the propagation structure inside a GCN when noise is added. More importantly, in our problem setting, the attacker does not rely on the edge information (i.e., the adjacency matrix).

A natural method to prevent the leakage of private information is to add noise, and one promising example is the application of differential privacy (DP) in deep learning. However, training a privacy-preserving GNN is more challenging than other privacy deep learning models due to the relational characteristics of graphs. There are few attempts to provide privacy protection in the field of graph-based learning algorithms. In [13], Hu et al. proposed a privacy-preserving graph learning framework (named DP-GCN) for classifying unlabeled nodes with non-sensitive latent representations. Its goal is to prevent non-private users from disclosing sensitive information about private users. In addition, Igamberdiev and Habernal [14] applied differentially-private gradient-based training to GCN by injecting DP noise into the model gradients. This method can effectively mitigate the leakage of personal sensitive information in text classification. Considering that existing DP methods require a trusted curator, these methods may not be suitable for practical graph learning services, such as Machine-Learning-as-a-Service (MLaaS) provided by Google.

Several studies have explored the possibilities of applying local differential privacy (LDP) to GNN models. For example, Hidano and Murakami [30] proposed DPRR, an LDP-based GNN framework that provides LDP for graph edges. The framework employs a random response method to perturb the graph structure, and preserves degree information through a strategic edge sampling approach. This study focuses on edge-local LDP, which may be insufficient in some highly sensitive scenarios. Sajadmanesh and Gatica-Perez [4] developed a node-level local private GNN that provides LDP for graph vertices. Similarly, Lin et al. [16] proposed a privacy-preserving GNN framework, which provides a decentralized graph analysis under edge-LDP. Tran et al. [31] proposed a heterogeneous randomized response method to perturb vertex features and graph structure. However, most existing LDP-based methods only discuss how to infer user identities from learning models (i.e., member inference attacks), and cannot be directly used to defend against the two privacy attacks proposed in this paper. Our problem is different from and more challenging than existing privacy-preserving GNNs.

Graph augmentation involves modifying the graph structure or generating new features, similar to data augmentation techniques in computer vision (CV) and natural language processing (NLP). Relatively little work has ivestigated graph augmentation. This is primarily attributed to the intricate non-Euclidean structure of graphs, which limits the possible manipulation operations. A similar work by Rong et al. [32] only considered randomly removing a fraction of edges during GNN training, which is similar to the Drop operation. In another work [33], the authors considered removing "noisy" edges and adding "missing" edges to improve the GNN's performance. The most relevant work to us is [34], which generates neighborhood features by a generative model conditioned on graph structure and node features. However, these methods aim to improve model performance rather than to preserve privacy in GNNs. Building upon recent developments in graph augmentation, we integrate the local graph augmentation (LGA) into our framework, which expands the local neighborhood by generating vertex augmentations, serving as a denoising mechanism to enhance the expressiveness of the learned model. Specifically, we present an inspiring finding that our approach not only improves model performance but also preserves user privacy.

III. Preliminaries

A. Problem Definition

In this paper, we denote a directed graph as $G = (\mathcal{V}, \mathcal{E}, X)$ where $\mathcal{V}$ and $\mathcal{E}$ represent the set of vertices and edges, respectively. $X \in \mathbb{R}^{|\mathcal{V}| \times d}$ is a feature matrix, where each vertex $v_i \in \mathcal{V}$ has a d-dimensional feature vector (i.e., $\mathbf{x}_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,d}\}$) with a corresponding label $y_i \in \{0, 1\}$. The task is to train a GNN model that outputs a class probability $\tilde{y}_i$ for each vertex.

B. Graph Convolutional Networks

The GNN model uses a set of stacked graph convolutional layers to perform effective information propagation on the graph, which takes a tuple $(\mathcal{A}, X)$ as the input, where $\mathcal{A}$ and X are the adjacency matrix and the feature matrix, respectively. Then, the GCN learns the vertex hidden representation of each vertex by aggregating the vectors of its adjacent neighbors. More formally, the graph structure and vertex features can be encoded by the neighborhood aggregation operation Agg() and update operation $\Theta()$:

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

$$h^l_v(\theta) = \Theta\left(\mathbf{h}^l_{\mathcal{N}(v)} \cdot W^l\right), \tag{2}$$

where h^l_v represents the hidden representation of a vertex v at layer l, $\oplus$ represents the merge operation. $\mathcal{N}(v)$ represents the set of neighbors of the vertex v, and W represents the weight matrix. Specifically, the Agg() is an aggregate function with

invariant permutation, such as max, sum, or mean. Each layer updates all its hidden neurons $H^l = \{h_v^l\}_0^n$ by an activation function Θ (such as Sigmoid and ReLU), where n represents the number of hidden neurons.

C. Local Differential Privacy

LDP uses a specific random perturbation mechanism M to perturb the user's real data record and sends the perturbed version to the data curator. To protect privacy, we require M to satisfy ϵ-LDP. Formally, we give the definition of LDP as follows:

Definition 1 (ϵ-LDP): A randomized mechanism M satisfies ϵ-LDP if and only if for any two input tuples x and x', and for any output $y \in Range(M)$, we have:

$$\Pr\left[M(x) = y\right] \leq \exp(\epsilon) \cdot \Pr\left[M\left(x'\right) = y\right]$$

where $\Pr[\cdot]$ denotes the probability. The privacy budget ϵ is a metric of privacy loss that controls the privacy-utility trade-off, i.e., smaller values of ϵ indicate higher privacy guarantees but lower utility. The LDP can provide users with plausible deniability.

IV. Attack Model

This section focuses on understanding the risk of privacy leakage under decentralized graph learning. Section IV-A briefly describes the interaction model between data holders. Then in Section IV-B and IV-C, we introduce the threat model and adversary model of the proposed LGA-PGNN in terms of its attack surface, the capabilities of the adversary and its goals.

A. Interaction Model Between Data Holders

Modeling complex graph-structured data using GNN models has been gradually advanced to many domains such as social network, recommendation systems, etc. GNN aims to learn a low-dimensional representation of network vertices while preserving network topology structure and vertex content. For example, in a GNN model, the computation of vertices is regulated by the information passed from their neighbor vertices. However, in a graph isolation setting, the collection and analysis of graph data could lead to serious privacy concerns, as vertex and interactions between them often implicitly contain user-sensitive information that can be exploited for malicious purposes.

Now, we present a real-world example. In many real-world applications, companies have established their own business graph datasets for training a large-scale GNN model and providing users with various commercial machine learning services, known as Machine-Learning-as-a-Service (MLaaS). Then, users send their local graphs to the cloud via an inference API, which returns the corresponding predictions from the trained GNN model. In addition, these collected local graphs can also be used to update the GNN model. A GNN trained on large-scale data will contain more data distributions, thereby improving its generalization ability.

In this case, data holders upload their local (private) graphs to the cloud, which then builds a GNN model to perform

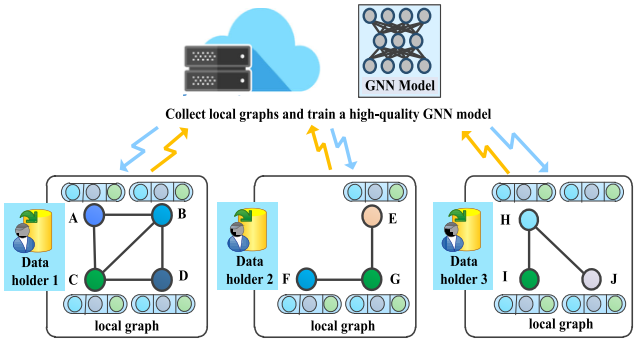


Fig. 1. Decentralized local graphs.

training and inference. To better understand this, an example is given in Fig. 1. The isolated data holders own parts of the whole graph. Let $G = \{\mathcal{G}^{k_1}, \mathcal{G}^{k_2}, \ldots, \mathcal{G}^{k_l}\}$ be a set of local graphs for k_n data holders $K = \{k_1, k_2, \ldots, k_n\}$. Each data holder k_i has a local graph $\mathcal{G}^{k_i} = (\mathcal{V}^{k_i}, X^{k_i}, \mathcal{A}^{k_i})$, where $\mathcal{V}^{k_i} = \{v_1^{k_i}, v_2^{k_i}, \ldots, v_{|\mathcal{V}|}^{k_i}\} \in \mathcal{V}$ is the set of vertices, $X^{k_i} = \{\mathbf{x}_1^{k_i}, \mathbf{x}_2^{k_i}, \ldots, \mathbf{x}_{|\mathcal{V}|}^{k_i}\}$ is the feature matrix, and $\mathcal{A}^{k_i}$ is the corresponding adjacency matrix. Formally, we define the interaction between the cloud and the data holders as follows.

1) GNN Training: In the training stage, each data holder k_i sends its feature matrix X^{k_i} with the corresponding labels $Y^{k_i} = \{y_1^{k_i}, y_2^{k_i}, \ldots, y_n^{k_i}\}$, $y_j^{k_i} \in C$ to the cloud, where C is the number of classes in our task. Then, the cloud performs the secure entity alignment to align vertices, which can compute the intersection of two sets without exposing those that are not in the intersection [1], [2]. Since this issue has been researched extensively in the prior work [1], [2], in this paper we assume that the cloud has aligned vertices. Based on the vertex connection information, the cloud trains a privacy-preserving GNN.

2) GNN Inference: In the inference stage, the cloud releases the trained GNN model as a black-box API $GNN_{API}()$ to the users. After that, the user sends its feature matrix $X^{\mathcal{V}^{(I)}}$ to the API, where $\mathcal{V}^{(I)} \in \mathcal{V}$ is a set of inference vertices. Following the standard MLaaS, the cloud uses the trained GNN API $GNN_{API}()$ to make inference on $X^{\mathcal{V}^{(I)}}$, and returns the prediction matrix $\mathbf{P}^{(I)}$ to the user. For simplicity, we consider only a single layer of GNN. This inference process can be described as follows:

$$\mathbf{h}_{\mathcal{N}}(\mathbf{x}_i^{v_i}, \mathcal{N}(\mathbf{x}_i^{v_i})) = \text{Agg}\left(\left\{\mathbf{x}_u^{v_i}, \forall u \in \mathcal{N}(v_i)\right\}\right) \quad (3)$$

$$GNN_{API}(\mathbf{x}_i^{v_i}, \mathbf{h}_{\mathcal{N}}) = \left(\mathbf{x}_i^{v_i} \oplus \mathbf{h}_{\mathcal{N}(v_i)}\right) W \quad (4)$$

$$\mathbf{p}_{v_i} = GNN_{API}(\mathbf{x}_i^{v_i}, \mathcal{N}(v_i)) \quad (5)$$

$$\mathbf{P}^{(I)} = \{\mathbf{p}_{v_1}, \mathbf{p}_{v_2}, \ldots, \mathbf{p}_{v_n}\} \quad (6)$$

where each class probability $\mathbf{p}_{v_i} \in \mathbb{R}^{1\times C}$ corresponds to a vertex v_i in $\mathcal{V}^{(I)}$, which consists of a vector $\{z_{i,1}, z_{i,2}, \ldots, z_{i,C}\}$ where each value z_{ic} in this class probability $\mathbf{p}_{v_i}$ corresponds to a confidence for a class $c \in C$.

B. Threat Model

Undoubtedly, the graph data isolation complicates the process of training an effective GNN model. In particular, insecure communication between the cloud and data holders greatly

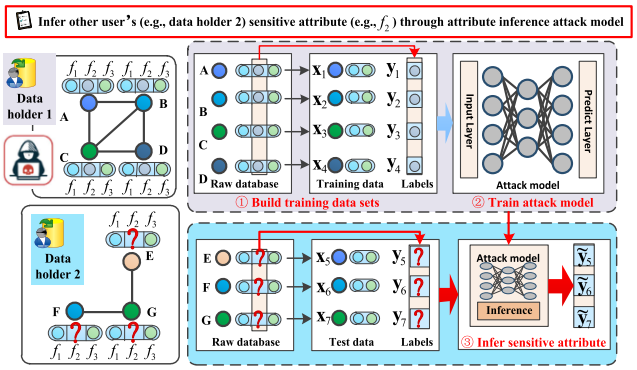


Fig. 2. An example of attribute inference attack.

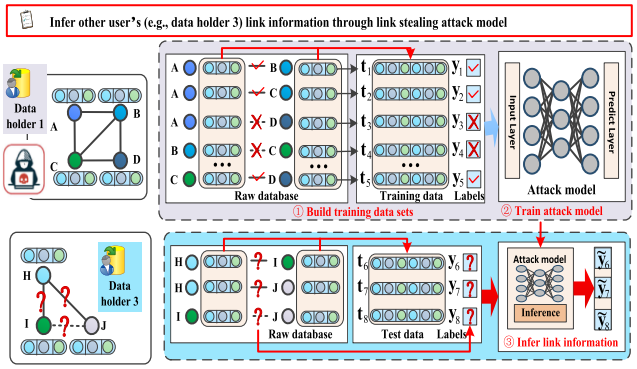


Fig. 3. An example of link stealing attack.

expands the attack surface during the collection and analysis of graph data. Based on the goals of the attacks, we list the potential risks of privacy leakage below.

1) Content Disclosure Risks: Generally, vertex attributes may contain personal information about the user, such as gender, age and occupation. If an adversary can directly attribute the compromised vertex attributes to a specific person, it will cause the leakage of user privacy. Therefore, vertex attributes must be safeguarded.

2) Edge Disclosure Risks: In a social graph, edges represent relationships between users. Edge information is considered private. In fact, users prefer to keep such information strictly confidential. For example, Alice is an HIV specialist who communicates frequently with patient Bob. When the doctor-patient relationship is exposed, the attacker can confidently infer that the patient Bob is likely to be infected with HIV. It requires a stricter protection of edge information.

C. Adversary Model and Assumptions

As mentioned earlier, the GNN model updates the central vertex's embeddings through message passing between neighboring vertices. Generally, an adversary wants to infer private information about the user, e.g., vertex attributes or link (edge) information in a social graph. Attacks can occur in two phases: model training and model inference. Based on the adversary's ability/knowledge, we distinguish between two types of adversaries, Adversary 1 and Adversary 2. We show the details of the two attacks in Fig. 2 and Fig. 3, respectively.

1) Attribute Inference Attack: We focus on limited-knowledge attacks where Adversary 1 has no knowledge about the classification model and its model weights, but owns a portion of the data of the entire training set (i.e., the local graph $\mathcal{G}^{(I)} = \{\mathcal{V}^{(I)}, \mathcal{E}^{(I)}, X^{(I)}\}$) through trading, crawling the network, or using other resources. In other words, the Adversary 1 can observe the partial vertex attributes $X^{(I)}$ (i.e., f_1 and f_3 in Fig 2). These partial vertex attributes are used to train an attack model Ψ_{AIAtk} and infer other users' private attributes $X^{(P)}$ (i.e., f_2 in Fig 2). In our case, the private attribute $X^{(P)}$ can be viewed as a label, i.e., $X^{(P)} = Y^{(P)}$. The gradient ∇_{AIAtk} of the attack model's loss can be computed as follows:

$$\begin{aligned}\nabla_{AIAtk} := \nabla^{(I)}_{X^{(I)}} \min \mathcal{L}_{\text{AIAtk}} \left(\Psi^{AIAtk}_{\theta^*}\left(\mathbf{x}_i^{(I)}, y_i^{(P)}\right)\right)\\ \text{s.t.} \quad \theta^* = \arg\min_{\theta} \mathcal{L}_{\text{train}} \left(\Psi^{AIAtk}_{\theta}\left(\mathbf{x}_i^{(I)}, y_i^{(P)}\right)\right)\end{aligned} \tag{7}$$

where $\mathbf{x}_i^{(I)} \in X^{(I)}$ is a feature vector in the training set $X^{(I)}$ held by the Adversary 1, and $y_i^{(P)} \in Y^{(P)}$ is the private attribute of the i-th vertex. The model parameter θ can be learned by minimizing the loss function $\mathcal{L}_{\text{AIAtk}}$. In the inference stage, the attack model can infer the users' private attributes by giving the optimal predictions $\tilde{y}_i^{(P)}$:

$$\tilde{y}_i^{(P)} = \Psi^{AIAtk}(\mathbf{x}_i^{(I)}) \tag{8}$$

2) Link Stealing Attack: Similar to Adversary 1, Adversary 2 does not know the details of the model, but has a local graph where all its vertex attributes and edges can be observed. Based on the link stealing attack model Ψ^{AIAtk}, Adversary 2 performs a link stealing attack to reveal private links between vertex pairs. The attack model Ψ^{AIAtk} is trained over a set of pairwise vertices, i.e., $X^{(I)} = \{\mathbf{x}_i^{(I)}, \mathbf{x}_j^{(I)}, e(\mathbf{x}_i^{(I)}, \mathbf{x}_j^{(I)})\}_0^{\mathcal{E}^{(I)}}$ where $\mathbf{x}_i^{(I)}$ and $\mathbf{x}_j^{(I)}$ can be any two vertices connected by an edge $e(\mathbf{x}_i^{(I)}, \mathbf{x}_j^{(I)}) \in \mathcal{E}^{(I)}$ in the local graph. Let $t_i^{(I)} =< \mathbf{t}_i^{(I)}, y_{e(\mathbf{x}_i^{(I)}, \mathbf{x}_j^{(I)})} >$ be a triple, where two vertex features $\mathbf{x}_i^{(I)}$ and $\mathbf{x}_j^{(I)}$ are concatenated together as a training data $\mathbf{t}_i^{(I)} = con(\mathbf{x}_i^{(I)}, \mathbf{x}_j^{(I)})$, and the edge $e(\mathbf{x}_i^{(I)}, \mathbf{x}_j^{(I)})$ between them as the corresponding label $y_{e(\mathbf{x}_i^{(I)}, \mathbf{x}_j^{(I)})}$. In addition to the "concatenation" method, we can also use "plus", "multiply" and "average" methods to construct the training dataset for the link stealing attack. Next, the attack model's gradient ∇_{LSAtk} can be computed as follows:

$$\begin{aligned}\nabla_{LSAtk} := \nabla^{(I)}_{X^{(I)}} \min_{\{\mathbf{x}_i^{(I)}, \mathbf{x}_j^{(I)}\} \in X^{(I)}} \mathcal{L}_{\text{LSAtk}} \left(\Psi^{LSAtk}_{\theta^*}(\mathbf{t}_i^{(I)})\right)\\ \text{s.t.} \quad \theta^* = \arg\min_{\theta} \quad \mathcal{L}_{\text{train}} \left(\Psi^{LSAtk}_{\theta}(\mathbf{t}_i^{(I)})\right).\end{aligned} \tag{9}$$

In the inference stage, the attack model can observe the private links between a pair of vertices $\mathbf{t}_i^{(I)} =< \mathbf{x}_i, \mathbf{x}_j >$ as follows:

$$\tilde{y}_{e(\mathbf{x}_i^{(I)}, \mathbf{x}_j^{(I)})} = e(\mathbf{x}_i^{(I)}, \mathbf{x}_j^{(I)}) = \Psi^{LSAtk}(\mathbf{t}_i^{(I)}) \tag{10}$$

V. Local Graph Perturbation for GNNs

In this paper, we propose LGA-PGNN, which encodes the structure and vertex features of the graph while protecting

the privacy of each individual user. In Section V-A, we first present the procedure of local graph perturbation with formal privacy guarantees. In Section V-B, we develop an LGA mechanism to enhance the vertex representation of the local graph. In Section V-C, we present how to construct a privacy graph convolutional layer in a GNN. Finally, Section V-D gives the correctness and applicability of the LGA-PGNN.

A. Collecting Multiple Vertex Attributes With LDP

In this subsection, we apply the LDP algorithm to the problem of graph isolation. Let us consider a scenario in which there are k_n geographically separated users, where each user k_i has a local graph $\mathcal{G}^{k_i} = (\mathcal{V}^{k_i}, X^{k_i}, \mathcal{A}^{k_i})$. Each row $\mathbf{x}_i^{k_i}$ of the feature matrix X^{k_i} contains a private d-dimensional numerical attributes $\{x_{i,1}, x_{i,2}, \ldots, x_{i,d}\}$. Then, a piecewise mechanism (PM) [35] is introduced to provide LDP protection to the user's data records. The user k_i perturbs the vertex attribute $x_{i,j}$ to output a perturbed value $x_{i,j}^*$. The input domain of the perturbed vertex attribute $x_{i,j}^*$ is in the range $[-\frac{e^{\epsilon/2}+1}{e^{\epsilon/2}-1}, \frac{e^{\epsilon/2}+1}{e^{\epsilon/2}-1}]$. Specifically, the noise value is actually a random variable drawn from a piece-wise constant probability distribution, with the following probability density function:

$$pdf\left(x_i^* = t \mid x_i\right) = \begin{cases} \dfrac{e^{\epsilon} - e^{\epsilon/2}}{2e^{3\epsilon/2} + 2e^{2\epsilon}}, & \text{if } t \in \left[-\dfrac{e^{\epsilon/2}+1}{e^{\epsilon/2}-1}, \alpha\right] \\ \dfrac{e^{\epsilon} - e^{\epsilon/2}}{2e^{\epsilon/2} + 2}, & \text{if } t \in [\alpha, \beta] \\ \dfrac{e^{\epsilon} - e^{\epsilon/2}}{2e^{3\epsilon/2} + 2e^{2\epsilon}}, & \text{if } t \in \left[\beta, \dfrac{e^{\epsilon/2}+1}{e^{\epsilon/2}-1}\right] \end{cases} \tag{11}$$

where $\alpha = \frac{e^{\epsilon/2}}{e^{\epsilon/2}-1} \cdot x_i - \frac{1}{e^{\epsilon/2}-1}$, and $\beta = \alpha + \frac{2}{e^{\epsilon/2}-1}$. After that, the user k_i sends the perturbed vertex attribute $x_{i,j}^*$ to the cloud. The PM outputs three piece-wise constant probability distributions, where the length and position of the "pieces" depend on the input data. Algorithm 1 (Line 1-7) presents the pseudo-code of the multi-attribute perturbation mechanism for multi-dimensional numerical data. Now we give the theoretical guarantees for the Algorithm 1 by the following lemmas.

Lemma 1: Algorithm 1 preserves ϵ-LDP.

Proof 1: Let $x_i^* = \mathcal{M}(x)$ be an perturbed value, which satisfies the LDP notion as shown in definition 1. For any two input vertex attributes x and x^*, we have

$$\frac{pdf\left(x^* \mid x\right)}{pdf\left(x^* \mid x'\right)} \leq \frac{\frac{e^{\epsilon} - e^{\epsilon/2}}{2e^{\epsilon/2}+2}}{\frac{e^{\epsilon} - e^{\epsilon/2}}{2e^{3\epsilon/2}+2e^{2\epsilon}}} = \exp(\epsilon)$$

Lemma 2: The piecewise mechanism is unbiased. For any vertex $v_i \in \mathcal{V}$ and any dimension $j \in \{1, 2, \ldots, d\}$, we have that $E[x_{i,j}^*] = x_{i,j}$.

Proof 2: Wang et al. [35] prove that $\mathbb{E}\left[x_{i,j}^*\right] = x_{i,j}$. This indicates that $x_{i,j}^*$ is an unbiased estimator of the input value $x_{i,j}$.

To achieve LDP, the baseline approach is to assign the same privacy budget ϵ/d to each attribute using a single attribute perturbation mechanism that satisfies $\frac{\epsilon}{d}$-LDP for each dimension. Then, according to the composition theorem [36], the total privacy budget satisfies ϵ-LDP, where the total privacy budget ϵ is shared among all dimensions of the high-dimensional input, i.e., $\epsilon = \sum_{j=1}^{d} \frac{\epsilon}{d}$. However, this solution could lead to a dramatic drop in the utility of the data. Consider that the amount of noise in each attribute is $O\left(\frac{\sqrt{d \log d}}{\epsilon\sqrt{n}}\right)$, which is correlated with the dimension d, i.e., when the dimension d is large, more noise will be incurred. This degrades the final accuracy significantly. Instead of perturbing all dimensions, we try to balance the privacy-accuracy trade-off by reducing the number of dimensions that need to be perturbed. In our case, the perturbation mechanism randomly perturbs m out of d dimensions. As a result, the privacy budget for each dimension increases from ϵ/d to ϵ/m, which implies that the noise variance is reduced.

Moreover, the variance of $x_{i,j}^*$ can affect the estimation accuracy, which is described as:

$$\mathrm{Var}\left[x_{i,j}^*\right] = \frac{d\left(e^{\epsilon/(2m)} + 3\right)}{3m\left(e^{\epsilon/(2m)} - 1\right)^2} + \left[\frac{d \cdot e^{\epsilon/(2m)}}{m\left(e^{\epsilon/(2m)} - 1\right)} - 1\right] \cdot x_{i,j}^2. \tag{12}$$

A lower variance will give a more reasonably accurate estimate.

In particular, to achieve minimum variance in estimating the mean, we optimize the sampling parameter m that controls how many dimensions are perturbed. We then have the following lemma:

Algorithm 1 LGA-PGNN

1: **User's side:**
2: #Inject random noise into the vertex attributes;
3: **for** $\mathbf{x}_i \in X,\ j \in [0, d]$ **do**
4: #Perturb m vertex attributes;
5: $x_{i,j}^* = Perturb(x_{i,j})$ (Perturb j-th vertex attribute $x_{i,j}$ via Eq. 11);
6: #Send the Perturbed vertex attributes $x_{i,j}^*$ to cloud;
7: **end for**
8: **Cloud's side:**
9: #Obtain all vertex representations X^*;
10: **for** each $epoch$ **do**
11: **for** $\mathbf{x}_v^* \in X^*$ **do**
12: #Perform the graph augmentation;
13: **for** $\mathbf{x}_s^* \in X_S^*$ **do**
14: $\mathbf{x}_s^* = CGAE_{API}(\mathbf{x}_s^*)$;
15: **end for**
16: **end for**
17: **for** $\mathbf{x}_i^* \in X^*, i \in [0, |V|], j \in [0, d]$ **do**
18: $\overline{h}(x_{i,j}^*) = \mathrm{Agg}\left(\left\{x_{u,j}^*, \forall u \in \mathcal{N}(v_i)\right\}\right)$;
19: **end for**
20: #Construct the PGC layer:
21: $\overline{h}_i(W) = \sum_{j \in d}\left(\overline{h}(x_{i,j}^*) \cdot W_{i,j}^T\right)$;
22: $\overline{P}_0(W) = \left\{\Theta\left(\overline{h}_i(W)\right)\right\}_{\overline{h}_i \in \overline{P}_0}$;
23: #Optimize model parameters;
24: $\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} \sum_{i=1}^{n} f()$.
25: **end for**

Lemma 3: To minimize the variance of the perturbation mechanism, the optimal sampling parameter $m^\star$ is obtained as:

$$m^\star = \max\left\{1, \min\left\{d, \left\lfloor \frac{\epsilon}{2.42} \right\rfloor\right\}\right\}.$$

The complete proof is provided in Appendix-A.

B. Augmentated Local Graph

In this subsection, we present the motivation and design considerations for local graph augmentation (LGA). Data augmentation techniques have been widely used in computer vision and natural language processing to prevent over-fitting in the training of deep learning models, e.g., cropping, rotating, and flapping in image augmentation. However, existing data augmentation methods cannot be directly applied to graph data. A major obstacle is that in contrast to grid data (e.g., images or text), the structure of the graph is irregular. To address this problem, we propose an LGA algorithm based on conditional generative auto-encoder (CGAE) to enhance the decentralized local graphs held by different users. Algorithm 1 (Lines 8-16) presents the procedure of LGA. In our case, LGA can be used as a denoising mechanism to achieve higher training accuracy while providing a strong privacy guarantee.

1) Conditional Generative Auto-Encoder for Decentralized Local Graphs: Given a vertex $\mathbf{x}_v^*$ in the local graph $\mathcal{G}^{k_i}$, we randomly sample S vertices $X_S^* = \{\mathbf{x}_s^*, \forall s \in \mathcal{N}_L(\mathbf{x}_v^*)\}$ from a set of all adjacent vertices $\mathcal{N}_L(\mathbf{x}_v^*)$ of $\mathbf{x}_v^*$ to form S neighboring pairs $\{< \mathbf{x}_v^*, \mathbf{x}_1^* >, < \mathbf{x}_v^*, \mathbf{x}_2^* >, \ldots, < \mathbf{x}_v^*, \mathbf{x}_S^* >\}$, where L is the number of neighbors of the given vertex $\mathbf{x}_v^*$. Our goal is to generate an augmented feature representation for the given vertex $\mathbf{x}_v^*$, which can then used for subsequent model training and inference. Based on the homophily assumption that neighbouring vertices tend to have similar features, we calculate the average vector for each pair of neighbors $< \mathbf{x}_v^*, \mathbf{x}_s^* >$, as follows:

$$\mathbf{x}_s^* = Mean(\mathbf{x}_v^*, \mathbf{x}_s^*) \tag{13}$$

where $Mean()$ is an average function. After that, we use a CGAE to take $\mathbf{x}_s^*$ as the input and output an augmented feature representation $\mathbf{x}_s^*$ for the given vertex $\mathbf{x}_v^*$. Formally, this process can be described as:

$$\mathbf{x}_s^* = CGAE_{API}(\mathbf{x}_s^*); \tag{14}$$

where $CGAE_{API}$ is the conditional generative auto-encoder model. By doing this, the conditional generative auto-encoder learns the conditional distribution of the feature vectors of its neighboring vertices of $\mathbf{x}_v^*$. We expect the generated feature vector to be close to the original ones.

In fact, it is not trivial to calculate the true posterior distribution $p_\theta(\mathbf{z}|\mathbf{x}_s^*, \mathbf{x}_v^*)$ precisely. Let's assume that the true posterior $p_\theta(\mathbf{z}|\mathbf{x}_s^*, \mathbf{x}_v^*)$ takes on approximate Gaussian form. To allow for tractable parameter learning, we introduce a fixed-form posterior distribution $q_\phi(\mathbf{z}|\mathbf{x}_s^*, \mathbf{x}_v^*)$ as a recognition model, which approximates the true posterior distribution $p_\theta(\mathbf{z}|\mathbf{x}_s^*, \mathbf{x}_v^*)$ using Gaussian latent variables, where ϕ and θ are variational parameters and generative parameters, respectively. The recognition model takes a neighboring pair $< \mathbf{x}_v^*, \mathbf{x}_s^* >$ as input, and outputs a latent representation $\mathbf{z}$ from the prior distribution $p_\theta(\mathbf{z}|\mathbf{x}_v^*)$. From a coding theory perspective, we call the recognition model $q_\phi(\mathbf{z}|\mathbf{x}^*)$ as a probabilistic encoder. It uses the feature vector of $\mathbf{x}_v^*$ as the condition, and produces a Gaussian distribution over the possible values of $\mathbf{z}$. Then, a generation model $p_\theta(\mathbf{x}^*|\mathbf{x}_v^*, \mathbf{z})$ is used to reconstruct the input data point, which can be referred to as a probabilistic decoder.

2) The Variational Bound: Following the previous work [34], we derive a lower bound on the marginal likelihood of the model based on the variational principle, which can act as a surrogate objective function. The variational lower bound can be described as:

$$\begin{aligned}\log p_\theta(\mathbf{x}_s^* \mid \mathbf{x}_v^*) &= \int q_\phi\left(\mathbf{z} \mid \mathbf{x}_s^*, \mathbf{x}_v^*\right) \log \frac{p_\theta\left(\mathbf{x}_s^*, \mathbf{z} \mid \mathbf{x}_v^*\right)}{q_\phi\left(\mathbf{z} \mid \mathbf{x}_s^*, \mathbf{x}_v^*\right)} \mathrm{d}\mathbf{z} \\ &\quad + KL\left(q_\phi\left(\mathbf{z} \mid \mathbf{x}_s^*, \mathbf{x}_v^*\right) \| p_\theta\left(\mathbf{z} \mid \mathbf{x}_s^*, \mathbf{x}_v^*\right)\right) \\ &\geq \int q_\phi\left(\mathbf{z} \mid \mathbf{x}_s^*, \mathbf{x}_v^*\right) \log \frac{p_\theta\left(\mathbf{x}_s^*, \mathbf{z} \mid \mathbf{x}_v^*\right)}{q_\phi\left(\mathbf{z} \mid \mathbf{x}_s^*, \mathbf{x}_v^*\right)} \mathrm{d}\mathbf{z},\end{aligned} \tag{15}$$

where KL denotes the Kullback-Leibler (KL) divergence that measures the closeness of two distributions, and encourages the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x}_s^*, \mathbf{x}_v^*)$ to be close to the true posterior $p_\theta(\mathbf{z}|\mathbf{x}_v^*)$. Then, the evidence lower bound can be written as:

$$\begin{aligned}\mathcal{L}\left(\mathbf{x}_s^*, \mathbf{x}_v^*; \theta, \phi\right) &= -KL\left(q_\phi\left(\mathbf{z} \mid \mathbf{x}_s^*, \mathbf{x}_v^*\right) \| p_\theta\left(\mathbf{z} \mid \mathbf{x}_v^*\right)\right) \\ &\quad + \frac{1}{L}\sum_{l=1}^{L} \log p_\theta\left(\mathbf{x}_s^* \mid \mathbf{x}_v^*, \mathbf{z}^{(l)}\right),\end{aligned} \tag{16}$$

where $\mathbf{z}^{(l)} = g_\phi\left(\mathbf{x}_v^*, \mathbf{x}_s^*, \mathbf{e}^{(l)}\right)$, $g_\phi(.)$ is a differentiable transformation parameterized by ϕ, $\mathbf{e}^{(l)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is an auxiliary variable, and L is the number of neighbors of a given vertex $\mathbf{x}_v^*$. We can maximize the evidence lower bound function w.r.t. the variational parameters ϕ and generative parameters θ, which is the same as minimizing KL divergence. For a chosen transformation function $g_\phi(.)$, we map the data point $\mathbf{x}_u^*$ and the random noise vector $\mathbf{e}$ to the latent variables $\mathbf{z}$ drawn from the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x}_s^*, \mathbf{x}_v^*)$, i.e., $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}_s^*, \mathbf{x}_v^*)$. In this case, we can ensure that the approximate posterior distribution $q_\phi(\mathbf{z}|\mathbf{x}_s^*, \mathbf{x}_v^*)$ is as close as possible to the true posterior distribution $p_\theta(\mathbf{z}|\mathbf{x}_v^*)$. After that, a feature vertex is generated using the sample $\mathbf{z}$. Specifically, in the objective function $\mathcal{L}\left(\mathbf{x}_s^*, \mathbf{x}_v^*; \theta, \phi\right)$, the first term (i.e. the KL divergence) can be used as a regularizer w.r.t. the parameters ϕ, while the second term can be regarded as an expected negative reconstruction error. Finally, we construct the estimator of Eq. (16) as follow:

$$\begin{aligned}\mathcal{L}\left(\mathbf{x}_s^*, \mathbf{x}_v^*; \theta, \phi\right) &\simeq \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2\right) \\ &\quad + \frac{1}{L}\sum_{l=1}^{L} \log p_\theta\left(\mathbf{x}_s^* \mid \mathbf{x}_v^*, \mathbf{z}^{(l)}\right),\end{aligned} \tag{17}$$

where J defines the dimension of the latent variable $\mathbf{z}^{(l)}$. In the optimization process, the model performs error backpropagation, which calculates the gradients of the objective function

$\nabla_{\boldsymbol{\theta},\phi}\mathcal{L}\left(\mathbf{x}_u^*, \mathbf{x}_v^*; \theta, \phi\right)$ using stochastic optimization methods such as stochastic gradient descent (SGD) or Adagrad.

C. Private Graph Convolutional Layer

In this part, we construct a private graph convolutional (PGC) layer $\overline{P}_0$ that computes a noisy estimation for each vertex. Algorithm 1 (Lines 17-24) presents the pseudocode of the LGA-PGNN training process. After the cloud receives all perturbed data records from each user, our private GNN model takes these records as input. We construct an unbiased aggregator function in the PGC layer, which can be used as a denoising mechanism to average the LDP noise injected into the vertex attributes. Then, we have:

$$\overline{h}(x_{i,j}^*) = Agg\left(\left\{x_{u,j}^*, \forall u \in \mathcal{N}(v_i)\right\}\right). \tag{18}$$

where $\overline{h}(x_{i,j}^*)$ is the j-th hidden neuron in the first GNN layer, $\mathcal{N}(v_i)$ is the neighbor set of the vertex v_i, where v_i also belongs to $\mathcal{N}(v_i)$ due to the self-loop. $Agg()$ is the mean aggregator function, which can be treated as a weighted sum to compute a noisy estimation for each vertex. Next, we construct the PGC layer $\overline{P}_0$ as follows.

$$\overline{h}_i(W) = \sum_{j\in d}\left(\overline{h}(x_{i,j}^*)\cdot W_{i,j}^T\right), \tag{19}$$

$$\overline{P}_0(W) = \left\{\Theta\left(\overline{h}_i(W)\right)\right\}_{\overline{h}_i\in\overline{P}_0}. \tag{20}$$

The unbiased (noisy) version of the mean aggregator function suggests a private estimation procedure: perturb the input data according to the LDP notion as shown in definition 1, and then perform information propagation using GNN's message passing mechanism to output an average value as an estimate of the mean value for the corresponding attribute. The following lemma shows that the hidden neuron $\overline{h}_i(W)$ using the perturbed input data is an unbiased estimator of $h_i(W)$.

Lemma 4: $\overline{h}_i(W)$ is an unbiased statistical estimation for $h_i(W)$ in the GNN's first layer. For any vertex $v_i \in \mathcal{V}$ and any dimension $j \in \{1, 2, \ldots, d\}$, we have that $E[\overline{h}_i(W)] = h_i(W)$

The complete proof is provided in Appendix-B.

Now, we use a mean aggregator function to estimate the mean for each vertex v_i in the PGC layer. The following lemma shows the relationship between the neighborhood size $|\mathcal{N}(v_i)|$ and the computational estimation error of the mean aggregator function, which gives an accuracy guarantee of $\frac{1}{|\mathcal{N}(v_i)|}\sum_{u\in N(v_i)} x_{u,j}$. Next, we give the following lemma.

Lemma 5: Let $Agg()$ be the mean aggregator function in the PGC layer. With at least $1-\delta$ probability, for any vertex v_i, we have:

$$\max_{j\in\{1,\ldots,d\}}\left|\overline{h}(x_{i,j}^*) - h(x_{i,j})\right| = O\left(\frac{\sqrt{d\log(d/\delta)}}{\epsilon\sqrt{|\mathcal{N}(v_i)|}}\right) \tag{21}$$

The complete proof is provided in Appendix-C.

Moreover, we stack the hidden layers $H_1, \ldots, H_k$ on the top of $\overline{P}_0$. These hidden layers $H_1, \ldots, H_k$ rely only on the PGC layer $\overline{P}_0$ without looking at the original data, and thus the computations of $H_1, \ldots, H_k$ do not reveal any private information about the input data. Finally, we define the cost function as $f(\mathbf{x}_i, \boldsymbol{\omega})$, which is used to measure the difference between the original and predicted values of y_i. The goal of the GCN is to find the optimal parameter vector $\boldsymbol{\omega}$ to minimize a given cost function. We also give the definition of the optimal model parameter $\boldsymbol{\omega}^*$ as below.

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}}\sum_{i=1}^{n} f(\mathbf{x}_i, \boldsymbol{\omega}). \tag{22}$$

In the inference stage, the private GNN returns a prediction function, i.e., $\mathbf{p}(v_i) = \Theta((\mathbf{x}_i^{v_i} \oplus Agg\left(\left\{\mathbf{x}_u^{v_i}, \forall u \in \mathcal{N}(v_i)\right\}\right))W)$.

D. The Correctness and Applicability of LGA-PGNN

We summarize the key steps of the proposed LGA-PGNN. Algorithm 1 presents the procedure of local graph perturbation, which perturbs a user's local graph by enforcing LDP noise to vertex representations. According to Lemma 1, the Algorithm 1 preserves ϵ-LDP. However, there are a large number of low-degree vertices with only a few neighbors in real graphs. We use LGA to perform local neighborhood expansion on low-order vertices to generate augmented representations of neighborhood vertices, and facilitate subsequent model training and inference. LGA enables GNN to cancel injected LDP noise to yield a relatively good approximation of neighborhood aggregation, when PCG layer aggregates information from a sufficiently large set of neighborhood vertices for the central vertex. Specifically, users perturb their local private graphs and then send them to the cloud. These received local graphs are only used as the input of the PCG layer in LGA-PGNN. This process runs only once, preventing the server from recovering the user's private graph information. There is no additional information from the input data to be accessed. This ensures that the computation of the k hidden layers $H_1, \ldots, H_k$ above the PGC layer $\overline{P}_0(W)$ is differentially private.

VI. Performance Evaluation

In this section, we have two analysis objectives. First, we analyze the attack effectiveness of attribute inference attack and link stealing attack. Secondly, we conduct extensive experiments based on real-world datasets to evaluate the effectiveness of LGA-PGNN under varying privacy budgets.

A. Experimental Setup

1) Experimental Environments: We conducted the experiments on a Linux machine with an Nvidia Geforce RTX 3080 GPU of 11 GB memory, and an Intel(R) Xeon(R) Silver 4210 CPU with a processor speed of 2.20 GHz. All models were run on a Python platform. The software environment is based on the PyTorch Geometric Library that is an open-source software database tool developed by Google. DGL is a Deep Graph Library containing many state-of-the-art GNN benchmarks. Table II shows the development environments used in our experiments.

TABLE I
ATTRIBUTE INFERENCE ATTACK UNDER VARYING PRIVACY BUDGETS

	DeezerHU Dataset				DeezerCR Dataset				DeezerRO Dataset			
ϵ	Accuracy	P	R	F-score	Accuracy	P	R	F-score	Accuracy	P	R	F-score
$\epsilon = 0.01$	60.89	42.75	8.99	14.85	61.11	40.70	8.06	13.45	61.83	38.33	8.31	13.67
$\epsilon = 0.1$	61.54	36.90	1.87	3.56	61.11	36.64	5.04	8.87	62.47	36.66	4.53	8.06
$\epsilon = 1$	61.28	43.52	6.78	11.74	61.33	43.33	10.02	16.28	62.68	41.99	7.12	12.18
$\epsilon = 5$	82.36	80.56	70.54	75.22	81.59	80.58	67.08	73.22	79.92	80.57	58.93	68.07
$\epsilon = 10$	87.44	88.42	77.00	82.32	86.09	87.78	73.08	79.76	85.86	90.58	68.16	77.79
$\epsilon = 20$	87.89	90.03	76.58	82.76	87.28	92.09	72.30	81.01	86.32	91.00	69.19	78.61
$\epsilon = +\infty$	87.99	90.81	76.07	82.79	87.07	90.44	73.29	80.96	86.34	90.44	70.18	78.87

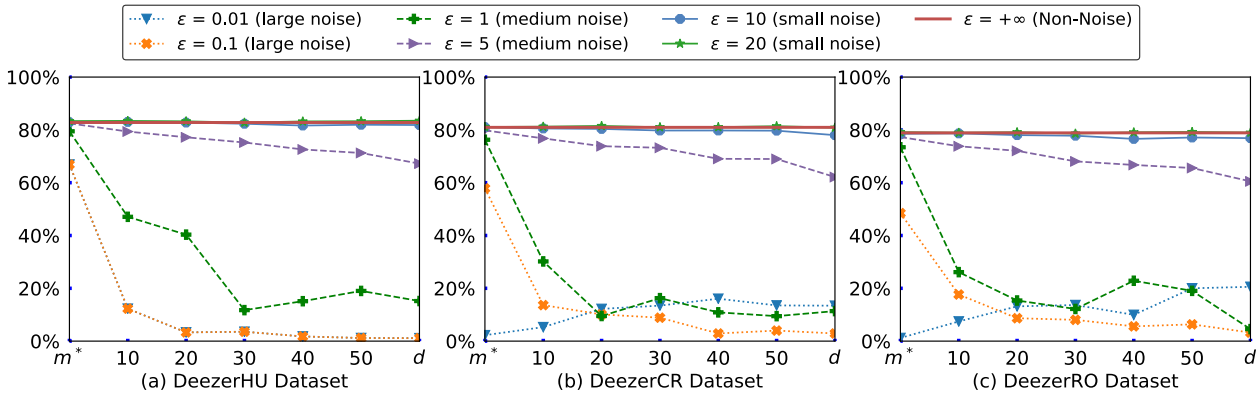


Fig. 4. F-scores of attribute inference attack under varying perturbation dimensions and privacy budgets.

TABLE II
THE MAIN DEVELOPMENT ENVIRONMENTS

Hardware	Information	Software	Version
GPU	Geforce RTX 3080 2.20GHz	CUDA	11.6
Memory	GPU 11GB, HDD 256GB	PyTorch	1.8.0
CPU	Intel Xeon Silver 4210	DGL	0.6.0

2) Parameters Setting: To implement the attribute inference attack and link stealing attack, we use a standard deep neural network as the attack model consisting of two fully connected layers with 128 neural units. Each layer is followed by a BatchNorm layer. Furthermore, in our experiments, the proposed LGA-PGNN uses a standard GNN model with two graph convolutional layers, followed by two fully connected layers with ReLU activation. Specifically, the learning rate is set to 0.01 and the dropout rate is set to 0.5.

B. Attribute Inference Attack Results

In the experiment, we perform attribute inference attacks. Table I shows the effect of attribute inference attacks under different privacy budgets ϵ. We can intuitively observe the performance changes of attribute inference attacks under varying privacy budgets from an attacker's perspective. In a non-private setting (i.e., $\epsilon = +\infty$), we measure the attacker's ability to predict the user's sensitive attributes. This means that the data holder does not take the LDP mechanism before uploading their own data to the server, i.e., no noise is added to the local data. In this setting, the attack F-score achieves the highest on the three datasets, and attackers can infer user private attributes with high confidence. However, when the privacy budget ϵ is very small (corresponding to a large amount of noise being added to the local data), the attack performs the worst on the three datasets. For example, when $\epsilon = 0.01$, the attack F-score dropped by 79.23% (from 82.79% to 3.56%) on the DeezerHU dataset, making the attack almost ineffective. Also, we can see that as the privacy budget ϵ increases gradually (corresponding to more loosed privacy protection), the attack performance achieves the best performance, and eventually achieves the same attack performance in the non-privacy setting. Specifically, when the privacy budget ϵ reaches 5, the attack F-score quickly rises to a plateau. It shows that our method is difficult to defend against the attack. Therefore, we propose setting the privacy budget ϵ as 1, which minimizes the attack performance while providing a strong privacy protection, with F-score decreasing from 82.79% to 11.75%.

In addition, we also show the trend changes of attack F-score under varying perturbation dimensions and privacy budgets. It can see that with the increase of the perturbation dimension, the attack F-score decreases continuously. Similar trends were observed for all privacy budgets. The downward trend is even more dramatic when privacy budgets are small. It is worth mentioning that the variance of the LDP mechanism greatly affects the estimation accuracy, i.e., a lower variance leads to a more accurate estimate. To reduce the noise variance, we randomly perturb m out of d dimensions and then the per-dimension privacy budget is increased from ϵ/d to ϵ/m. After theoretical analysis, the optimal sampling parameter is set to m^*. This shows that we theoretically only need to perturb m^* dimension to obtain a low variance. However, under this setting, our approach is incapable of defending against the attribute inference attack, as shown by the results in Fig. 4, where the attack F-score is almost optimal under all privacy budgets. In addition, when we perturb all dimensions d, the attack F-score is the worst under all privacy budgets.

We compare our method with several baseline methods, namely RandG-AIA, RI-MA, and FP-MA. For the attribute

TABLE III
LINK STEALING ATTACK UNDER VARYING PRIVACY BUDGETS

	Cora Dataset				Citeseer Dataset				Pubmed Dataset			
ϵ	Accuracy	P	R	F-score	Accuracy	P	R	F-score	Accuracy	P	R	F-score
ϵ = 0.01	50.22	51.86	20.79	29.68	49.75	49.96	31.96	38.98	49.84	48.81	28.34	35.86
ϵ = 0.1	49.35	49.85	40.51	44.70	50.24	50.66	35.82	41.97	50.40	49.80	32.02	38.98
ϵ = 1	49.21	49.69	41.49	45.22	49.40	49.73	70.66	58.38	49.89	49.16	37.43	42.50
ϵ = 5	49.09	49.51	38.41	43.26	49.00	47.45	14.32	22.00	49.88	49.52	67.29	57.05
ϵ = 10	52.86	53.15	56.59	54.82	54.18	62.58	21.80	32.34	61.27	58.90	71.84	64.73
ϵ = 20	90.60	90.41	91.04	90.73	90.31	89.40	91.57	90.47	91.88	89.94	94.12	91.98
ϵ = $+\infty$	93.82	92.05	96.08	94.02	92.50	90.74	94.74	92.70	93.85	92.30	95.54	93.89

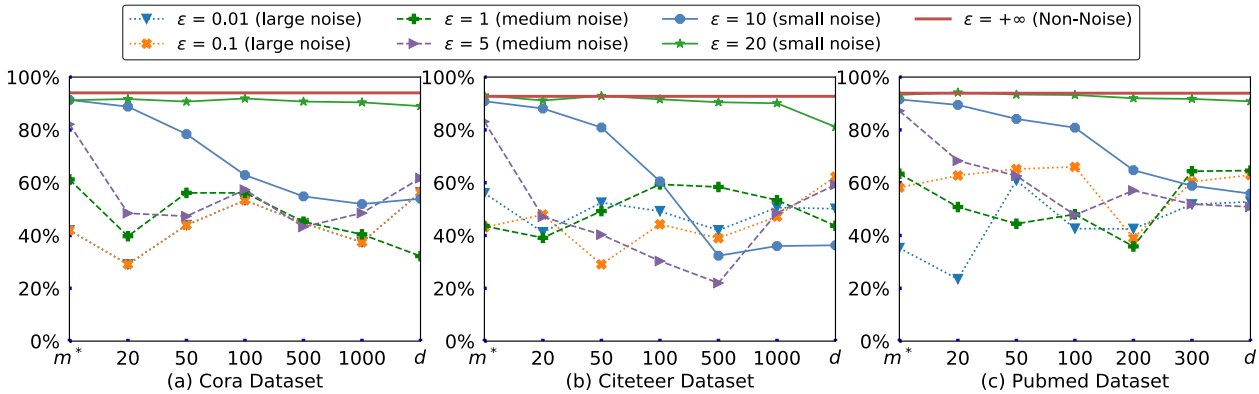


Fig. 5. F-scores of link stealing attack under varying perturbation dimensions and privacy budgets.

TABLE IV
ATTRIBUTE INFERENCE ATTACK PERFORMANCE AND BASELINES

Methods	DeezerHU		DeezerCR		DeezerRO	
	P	R	P	R	P	R
RAN-AIA	38.51	49.79	37.22	49.64	36.12	48.77
RI-MI	39.90	33.70	36.62	38.96	35.52	46.89
FP-MA	37.64	48.51	36.72	51.00	39.71	25.15
KNN-AIA	88.14	73.80	86.13	71.22	87.16	66.59
LR-AIA	87.90	73.02	88.96	68.66	89.68	65.85
RF-AIA	87.73	76.61	87.53	72.83	87.12	70.22
SVM-AIA	87.91	75.70	87.45	71.57	85.85	71.00
Our work	90.81	76.07	90.44	73.29	90.44	70.18

inference attack, we developed a baseline approach, RandG-AIA, in which an adversary randomly predicts user sensitive attributes. In [37], both RI-MA and FP-MA are used to reconstruct missing attributes. We re-implement the RI-MA and FP-MA methods in our experimental setting and use their architectures to infer user sensitive attributes. It is noteworthy that these baseline methods perform relatively poorly. In addition, we design several different types of attack models, including k-Nearest Neighbors (KNN), Logistic Regression (LR), Random Forests (RF), and Support Vector Machines (SVM), which are frequently employed in classification tasks. When the attacker is an LR-based model (LR-AIA), it outperforms the other baselines. However, our method consistently outperforms all comparative methods, which means that our method is more effective at inferring user sensitivity attributes. Table IV shows the accuracy of attribute inference attack for each attack model.

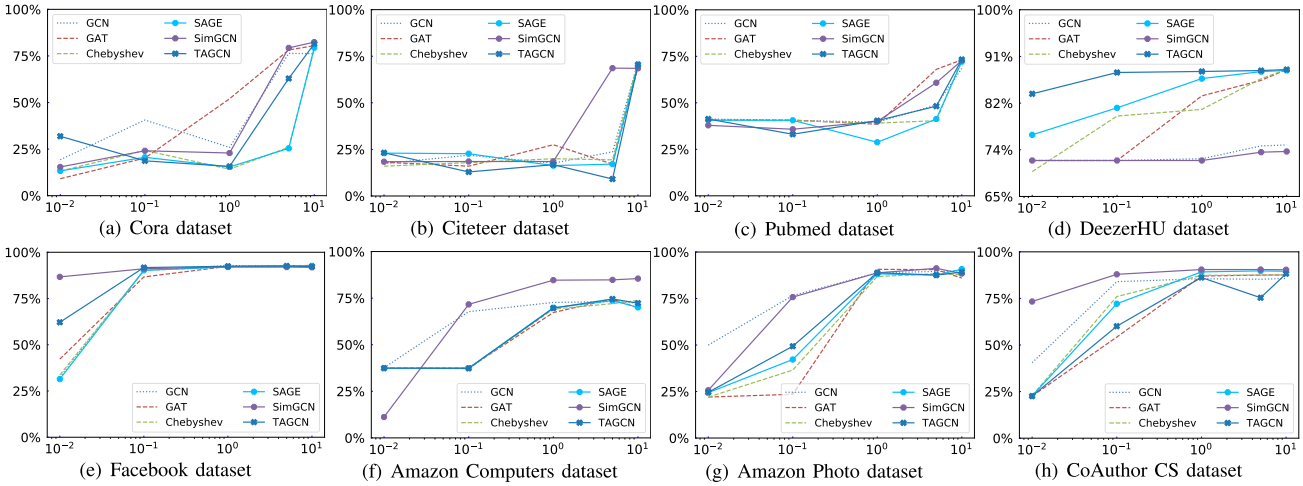
C. Link Stealing Attack Results

We intuitively observe the prediction ability of link stealing attack models to users' private link information. In Table III, we report the performance of the link stealing attack model with different privacy budgets ϵ. In a non-private setting (i.e., $\epsilon = +\infty$), F-score is the highest on all three datasets, such as 94.02% F-score on the Cora dataset. This means that users are very vulnerable to link stealing attacks. As expected, when the privacy budget is small, our privacy preserving method can significantly reduce the attack F-score. For example, when $\epsilon = 0.01$, the attack model shows the worst performance on the three datasets. As shown in Table III, for the Cora dataset, the attack F-score dropped by 64.34% (from 94.02% to 29.68%). The reason is that too much noise is added, which makes it difficult for the attack model to converge to the global optimum, so that it cannot predict the user's private link information correctly. When the privacy budget ϵ is set to 20, the attack model becomes stable. A larger ϵ means that less noise is added to the local data, which reduces the interference to the model training, and thus achieves better performance via gradient descent. In this case, it is difficult for us to defend against link stealing attacks. To make a privacy-accuracy trade-off, we recommend setting the privacy budget ϵ to less than or equal to 10. The experimental results clearly show that our approach not only reduces the performance of link stealing attacks, but also provides strong evidence in terms of improving privacy and utility.

Moreover, Fig. 5 provides a reference for selecting privacy budget value under different perturbation dimensions. As we know, a smaller privacy budget can provide stronger privacy protection, but less utility. When ϵ is less than or equal to 5, the attack model yields the low F-score on the three datasets. In this case, our approach is effective against such attacks. To achieve a trade-off between privacy and accuracy, we set the perturbation dimension as 500. The attack model yields the lowest F-score in Cora and Citeteer datasets, with a reduction of 50.76% (from 94.02% to 43.26%) and 70.70% (from 92.70% to 22.00%), respectively. When the perturbation

TABLE V
EFFECT OF LDP ON THE PERFORMANCE OF LGA-PGNN COMPARING TO NON-PRIVATE LGA-PGNN APPROACH

Model		Cora	Citeseer	Pubmed	Deezer	Facebook	Amazon Computers	Amazon Photo	CoAuthor CS
non-private LGA-PGNN	$\epsilon = +\infty$	81.61	72.20	76.80	74.63	92.21	82.60	89.57	89.00
LDP-GNN	$\epsilon = 0.01$	19.30	17.40	34.50	71.74	32.44	37.65	49.95	40.52
	$\epsilon = 0.1$	40.60	21.90	32.30	71.74	90.34	67.73	76.59	83.98
	$\epsilon = 1$	25.90	17.60	39.60	72.07	92.83	72.68	88.86	85.58
	$\epsilon = 5$	76.40	23.70	48.80	74.47	92.55	73.11	89.54	85.29
	$\epsilon = 10$	76.20	71.70	68.50	74.66	92.81	72.34	89.79	85.73
LGA-PGNN	$\epsilon = 0.01$	25.20	30.90	41.30	71.80	29.09	47.54	80.95	65.34
	$\epsilon = 0.1$	23.20	31.10	40.60	71.88	90.05	77.61	86.55	84.51
	$\epsilon = 1$	25.00	25.70	38.50	73.41	92.11	81.72	89.10	89.19
	$\epsilon = 5$	79.30	34.30	63.10	74.74	92.74	81.29	89.56	89.43
	$\epsilon = 10$	80.30	69.60	69.40	74.96	93.18	81.90	90.81	89.29



(a) Cora dataset (b) Citeteer dataset (c) Pubmed dataset (d) DeezerHU dataset
(e) Facebook dataset (f) Amazon Computers dataset (g) Amazon Photo dataset (h) CoAuthor CS dataset

Fig. 6. The effect of privacy budget ϵ on experimental results.

TABLE VI
LINK STEALING ATTACK PERFORMANCE AND BASELINES

Methods	Cora		Citeseer		Pubmed	
	P	R	P	R	P	R
RAN-LSA	50.29	50.90	49.90	50.74	49.58	50.34
LSA2	86.70	86.70	90.10	90.10	78.80	78.80
LSA2-EUC	40.03	20.88	35.54	38.25	33.67	31.81
LSA2-COS	79.87	59.95	89.62	69.95	88.68	66.72
KNN-LSA	77.81	95.22	66.49	85.48	77.81	95.22
LR-LSA	59.81	58.64	63.85	54.80	61.74	60.54
RF-LSA	91.77	93.47	89.01	91.52	91.13	94.19
SVM-LSA	62.39	63.03	65.63	61.99	61.66	61.75
Our work	92.05	96.08	90.74	94.74	92.30	95.54

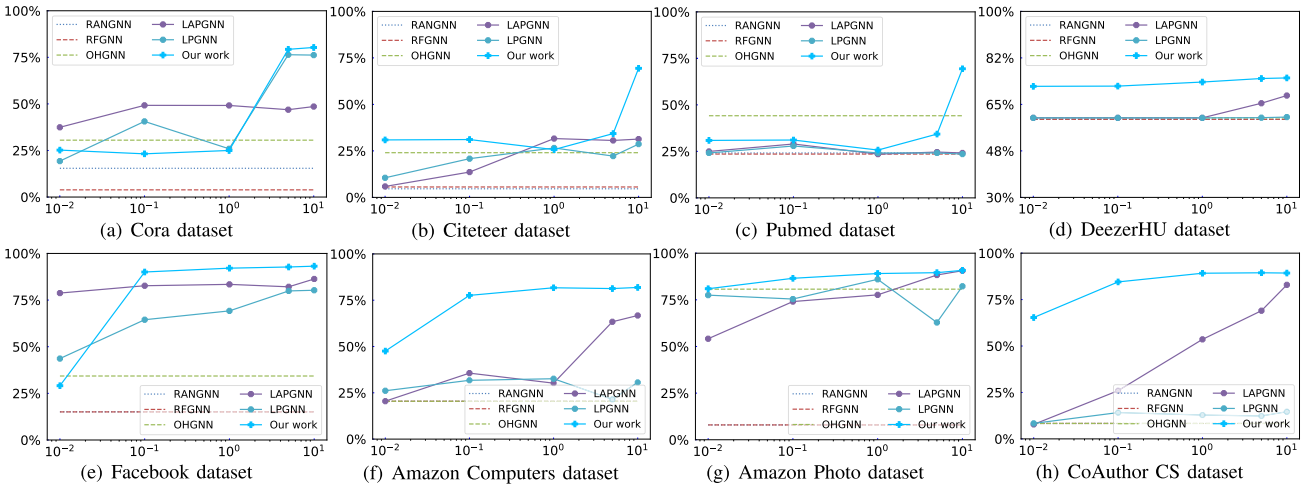
dimension is set to 100, the attack F-score is reduced by 46.14% (from 93.89% to 47.75%) on Pubmed dataset. The result shows that our method can resist link stealing attack effectively, which provides a strong guarantee in terms of privacy and practicality.

Table VI reports the experiment results (precision and recall) of our method and the baselines. RandG-LSA is an ideal baseline that randomly predicts sensitive links between nodes. Furthermore, we compare our model with the state-of-the-art LSA attack [27]. In this paper, the authors discuss several attack methods with different prior knowledge, among which their attack-2 is the closest to our scenario. In our study, we re-implement the attack-2 (namely LSA2), which calculates the correlation distance between the posteriors given by the target model to infer sensitive links between nodes of interest. In addition, [27] gives eight different distance metrics. Following this work, we use Euclidean distance and cosine distance (namely LSA2-EUC and LSA-COS) to calculate distances between node attributes, as they achieve optimal performance in almost all settings. We can observe that LSA2 performs slightly better than LSA2-COS on the Cora and Citeseer datasets, but poorly on the Pubmed dataset. Specifically, we explored a variety of neural networks as attack models, including KNN-LSA, LR-LSA, RF-LSA, and SVM-LSA. Among all baseline models, RF-LSA performs the best. However, it is inferior to our model in all cases. We can observe that the precision of our model is 92.05%, 90.74%, and 92.30% on the Cora, Citeseer, and Pubmed datasets, which significantly beats the best baseline system (RF-LSA) by 0.28%, 0.27% and 1.17%, respectively. This result proves the validity of our method.

D. Comparative Classification Performance

In this part, we evaluate the classification performance of LGA-PGNN under different privacy budgets. In our case, the server uses LGA-PGNN to provide users with various commercial machine learning services.

1) Non-LDP vs. LDP: To demonstrate the effectiveness of our model, we quantified the performance loss of LGA-PGNN against the non-private case (i.e., non-private LGA-PGNN). To have a fair comparison, the non-private LGA-PGNN adopts a standard two-layer GCN without any perturbation mechanism. Then, we compared the performance of LGA-PGNN with and without LDP in Table V. For example, the non-private LGA-PGNN (without LDP) gives accuracies of 81.61% on Cora, 72.20% on Citeteer, and 76.80% on Pubmed



(a) Cora dataset (b) Citeteer dataset (c) Pubmed dataset (d) DeezerHU dataset

(e) Facebook dataset (f) Amazon Computers dataset (g) Amazon Photo dataset (h) CoAuthor CS dataset

Fig. 7. F-score on eight datasets. wrt. privacy budget ϵ, with DP.

test sets. Specifically, to evaluate the effectiveness of the LDP mechanism, we provide a privacy-preserving baseline model LDP-GNN, which injects random noise into the input data. As anticipated, LDP-GNN decreases accuracy. For example, LGA-PGNN incurs significant errors on all datasets when the privacy budget is small (i.e., at $\epsilon = 0.01$). The reason is that too much noise is added, making it difficult for the model to learn useful information. The performance of LDP-PGNN improves as the privacy budget ϵ increases. For a very loose ϵ, we observe that the accuracy is almost identical to that of non-private LGA-PGNN. This phenomenon is consistent with the inescapable information theory of the privacy-utility trade-off. 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 LDP-GNN provides compelling evidence on improving privacy and utility.

2) Effects of Graph Augmentation: To demonstrate the effectiveness of the proposed LGA, we intuitively observed the performance gain of LGA-PGNN with respect to the private GNN with LDP mechanism (i.e., LDP-GNN) under different privacy budgets. As shown in Table V, although the LDP-GNN model uses the LDP mechanism to protect the user's private data, injecting random noise directly into the input data leads to a decrease in the utility of the model. Moreover, LGA-PGNN employed the graph augmentation to enhance the performance of the LGA-PGNN model. In this case, noise added by LDP mechanism can be eliminated partially. As we can be seen, the change trend of LGA-PGNN is always consistent with that of LDP-GNN. We show that in a graph augmentation setting, LGA-PGNN almost always outperforms LDP-GNN on all datasets under the same privacy level ϵ. For example, when $\epsilon = 10$, the accuracy of LGA-PGNN on Cora dataset is approximately 4% higher than that of LDP-GNN. This result proves that LGA-PGNN not only enhances the utility of the private GNN model, but also provides a private guarantee.

3) Privacy-Preserving GNN Models: To highlight the significance of this study, our privacy preserving method is generalized and applied in six popular graph neural networks. These baseline models include GCN [17], GAT [18], Chebyshev [19], SAGE [20], SimGCN [21], TAGCN [22]. Fig. 6 illustrates the accuracy of each model under different privacy budgets ϵ. We discuss the utility and privacy level of our privacy preserving method under different graph neural network architectures. It can be see that different graph neural network architectures exhibit different behaviors in terms of their tolerance for the amount of noise added to the input data. When ϵ is small, all baseline models gain worse performance. This is because a smaller size would incur more noise. As ϵ increases, the performance of all baseline models saturates at a certain point, and improves significantly on all benchmark datasets. In this case, the improved data utility results in more accurate learning of all baseline models, almost all of which are comparable to the no-privacy case. Overall, the results indicate that for all baseline models, our privacy preserving method allows for better privacy protection using smaller value of ϵ without sacrificing too much accuracy.

4) Comparison With Other State-of-the-Art Methods: To highlight the significance of this study, we implemented state-of-the-art LDP mechanism baselines. As these baselines target different problem settings, we modified them to achieve vertex-level and edge-level graph privacy protection, tailored specifically to our problem settings. These baseline methods are introduced below:

- RANGNN: The RANGNN [4] consists of four-layer GCN, which randomly initializes the input features with a Gaussian distribution (with a mean of 0.0 and standard deviation of 0.01), optimizing the model with stochastic gradient descent (SGD).
- RFGNN: A standard model of GCN was originally proposed by [17]. Similar to RANGNN, the RFGNN perturbs the input features at random as discussed by [4].
- OHTGNN: Following the previous work [4], [38], we use the one-hot encoding of vertex degrees in the input layer of the standard GCN, which can be regarded as a completely private method.
- LAPGNN: Following the settings of [4] and [39], we perturb the input features by adding Laplace noise [11], [12].
- LPGNN: The LPGNN was originally designed for vertex-level privacy. Following their approach [4], [16], we use

a multi-bit mechanism to privately collect vertex features while making it suitable for our problem setup.

We compare LGA-PGNN with these private-preserving techniques. Fig. 7 illustrates the F-score of each algorithm under different privacy budgets. We see that RANGCN, RFGNN and OHTGCN yield inferior performance. LPGNN performs considerably better than the two fully private methods (RANGCN and OHTGCN) but worse than LAPGNN. In addition, the baselines LPGNN and LAPGNN perform poorly on the Tor-nonTor dataset. This result shows that LPGNN and LAPGNN are easily misled due to a lack of an adequate sample size. Importantly, we observe that our LGA-PGNN outperforms all other private-preserving methods in all cases, demonstrating that our method can capture useful information. Specifically, our LGA-PGNN provides both edge-level and vertex-level privacy guarantees based on the application requirements.

VII. Conclusion

In this paper, we build a local graph augmentation-based private graph neural network (LGA-PGNN). In order to address the privacy issues pertaining to the graph data isolation, we designed a privacy graph convolutional layer with formal privacy guarantees, in which user's local graph information is kept private by injecting LDP noise into the local graphs held by different data holders. We performed local neighborhood expansion on low-degree vertices to enhance the expressiveness of the learned model. In fact, our privacy-preserving method can be easily generalized to other GNN models. We evaluated the performance of LGA-PGNN on real-world datasets for node classification. The results demonstrate that LGA-PGNN achieves high accuracy while providing a rigorous privacy guarantee. The fact that the superior performance of our method indicates that privacy-preserving graph learning is a worthwhile exploration.

This paper presents two privacy attacks for inferring private node attributes and links. Future research will focus on exploring new privacy attacks to recover private graph structure information, such as subgraphs. In addition, we will try to develop new LDP techniques to enhance privacy protection in distributed graph learning. In future research, we will also explore other graph augmentation mechanisms that are more effective than the proposed LGA.

Appendix

A. Proof of Lemma 3

Proof 3: We need to find the optimal sampling parameter m^* that minimizes the upperbound of the variance $\mathrm{Var}\left[x^*_{i,j}\right]$:

$$m^\star = \arg\min_{m} \max_{x^*_{i,j}} \mathrm{Var}\left[x^*_{i,j}\right] \tag{A. 1}$$

Recall that in Eq. (12), the variance of $x^*_{i,j}$ is given as follows.

$$\mathrm{Var}\left[x^*_{i,j}\right] = \frac{d\left(e^{\epsilon/(2m)}+3\right)}{3m\left(e^{\epsilon/(2m)}-1\right)^2} + \left[\frac{d\cdot e^{\epsilon/(2m)}}{m\left(e^{\epsilon/(2m)}-1\right)} - 1\right]\cdot x^2_{i,j}, \tag{A. 2}$$

where the input domain of $x_{i,j}$ is in the range $[-1, 1]$. Therefore, when $x_{i,j} = \pm 1$, the variance of $x^*_{i,j}$ is maximized. We give the worst-case variance of $x^*_{i,j}$ as follows.

$$\begin{aligned}\mathrm{WorstVar}\left[x^*_{i,j}\right] &= \max_{x^*_{i,j}} \mathrm{Var}\left[x^*_{i,j}\right] \\ &= \frac{d(e^{\epsilon/(2m)}+3)}{3m\cdot(e^{\epsilon/(2m)}-1)^2} + \frac{d\cdot e^{\epsilon/(2m)}}{m\left(e^{\epsilon/(2m)}-1\right)} - 1\end{aligned} \tag{A. 3}$$

Next, we set $z = \frac{\epsilon}{2m}$ and $C = \frac{2d}{\epsilon}$. Then, Eq. (A. 3) can be further derived as

$$\begin{aligned}\mathrm{WorstVar}\left[x^*_{i,j}\right] &= C\cdot z\cdot\frac{e^z+3}{3(e^z-1)^2} + C\cdot z\cdot\frac{e^z}{e^z-1} - 1 \\ &= C\cdot z\cdot\left(\frac{e^z+3}{3(e^z-1)^2} + \frac{e^z}{e^z-1}\right) - 1 \\ &= C\cdot z\cdot\left(\frac{3(e^z)^2-2e^z+3}{3(e^z-1)^2}\right) - 1\end{aligned} \tag{A. 4}$$

Specifically, minimizing Eq. (A. 3) with respect to m is equivalent to minimizing Eq. (A. 4) with respect to z. Let $z^\star$ to be the optimal z. We can recover $m^\star$ as $\frac{\epsilon}{2z^\star}$. Then, we have:

$$z^\star = \arg\min_{z}\left[C\cdot z\cdot\left(\frac{3(e^z)^2-2e^z+3}{3(e^z-1)^2}\right) - 1\right] \tag{A. 5}$$

Since the constants, C and -1, do not depend on z, we consider dropping them from Eq. (A. 4). After that, Eq. (A. 4) is re-writed as:

$$z^\star = \arg\min_{z}\left[z\cdot\frac{3(e^z)^2-2e^z+3}{3(e^z-1)^2}\right] \tag{A. 6}$$

Let $f(z) = z\cdot\frac{3(e^z)^2-2e^z+3}{3(e^z-1)^2}$ with respect to $z = \frac{\epsilon}{2m}$, which is a convex function. Then, we look for the minimum of $f(z)$ on $(0, +\infty)$ by taking the derivative of this function $f(z)$ with respect to z. The result is given by

$$\begin{aligned}f'(z) &= \frac{d}{dz}z\cdot\left[\frac{3(e^z)^2-2e^z+3}{3(e^z-1)^2}\right] \\ &= \left[\frac{3(e^z)^2-2e^z+3}{3(e^z-1)^2}\right] \\ &\quad + z\cdot\left[\frac{-12(e^z)^3+12(e^z)}{9(e^z-1)^4}\right] \\ &= \left[\frac{9(e^z)^4-24(e^z)^3+30(e^z)^2-24e^z+9}{9(e^z-1)^4}\right] \\ &\quad + z\cdot\left[\frac{-12(e^z)^3+12(e^z)}{9(e^z-1)^4}\right]\end{aligned} \tag{A. 7}$$

By setting $f'(z) = 0$, we have:

$$z = \frac{9(e^z)^4-24(e^z)^3+30(e^z)^2-24e^z+9}{12(e^z)^3-12(e^z)} \tag{A. 8}$$

After solving the above equation, the minimum of $f(z)$ is obtained, i.e., $z^\star \approx 1.2097$. Then, we have that $m^\star = \frac{\epsilon}{2z^\star} = \frac{\epsilon}{2.42}$. Finally, we specify m^* by

$$m^\star = \max\left\{1, \min\left\{d, \left\lfloor\frac{\epsilon}{2.42}\right\rfloor\right\}\right\}. \tag{A. 9}$$

B. Proof of Lemma 4

Proof 4: Based on Eqs. (18) and (19), we have:

$$E[\overline{h}_i(W)] = E\left[\sum_{j\in d}\overline{h}(x^*_{i,j})\cdot W^T_{i,j}\right] = E\left[\sum_{j\in d}\text{Agg}\left(\left\{x^*_{u,j}, \forall u\in\mathcal{N}(v_i)\right\}\right)\cdot W^T_{i,j}\right] \tag{B. 10}$$

Specifically, a hidden neuron $\overline{h}_i(W)$ defined by Eq. 19 at the PGC layer $\overline{P}_0(W)$ can be considered a weighted summation of the input vertex attributes. Therefore, this formula can be further derived as

$$E[\overline{h}_i(W)] = \sum_{j\in d}\text{Agg}\left(\left\{E[x^*_{u,j}], \forall u\in\mathcal{N}(v_i)\right\}\right)\cdot W^T_{i,j} \tag{B. 11}$$

Then, accordding to Lemma 2, we can derive the expectation of $\overline{h}_i(W)$ as below.

$$E[\overline{h}_i(W)] = \sum_{j\in d}\text{Agg}\left(\left\{x^*_{u,j}, \forall u\in\mathcal{N}(v_i)\right\}\right)\cdot W^T_{i,j} = \overline{h}_i(W). \tag{B. 12}$$

C. Proof of Lemma 5

Proof 5: By Lemma 2, we know that for any $i\in|\mathcal{V}|$, $x^*_{i,j}-x_{i,j}$ has zero mean, i.e.,

$$\mathbb{E}\left[x^*_{i,j}-x_{i,j}\right]=0. \tag{C. 13}$$

As mentioned in Lemma 3, we uniformly select m^* attributes from all d attributes of x_i, and set $x^*_{i,j}=\frac{d}{m}\cdot t_{i,j}$ with probability $\frac{m}{d}$ and 0 with probability $1-\frac{m}{d}$. Moreover, the input domain of a perturbed vertex attribute $x^*_{i,j}$ is in the range $[-\frac{e^{\epsilon/2}+1}{e^{\epsilon/2}-1}, \frac{e^{\epsilon/2}+1}{e^{\epsilon/2}-1}]$. Hence, we get:

$$|x^*_{i,j}-x_{i,j}|\le\frac{d}{m}\cdot\frac{e^{\epsilon/2}+1}{e^{\epsilon/2}-1} \tag{C. 14}$$

Given an mean aggregator function in the PGC layer, for any vertex $v_i\in\mathcal{V}$ and any dimension $j\in\{1,2,\ldots,d\}$, we have:

$$\begin{cases} h(x_{i,j}) = \dfrac{1}{|\mathcal{N}(v_i)|}\displaystyle\sum_{u\in N(v_i)} x_{u,j} \\ \overline{h}(x^*_{i,j}) = \dfrac{1}{|\mathcal{N}(v_i)|}\displaystyle\sum_{u\in\mathcal{N}(v_i)} x^*_{u,j} \end{cases} \tag{C. 15}$$

Considering Eqs. (C. 13), (C. 14) and (C. 15), and using the Bernstein's inequality, we have:

$$\Pr\left[\left|\overline{h}(x^*_{i,j})-h(x_{i,j})\right|\ge\lambda\right] = \Pr\left[\left|\sum_{i=1}^{n}\left\{x^*_{i,j}-x_{i,j}\right\}\right|\ge n\lambda\right] \le 2\cdot\exp\left(-\frac{(n\lambda)^2}{2\sum_{i=1}^{n}\text{Var}\left[x^*_{i,j}\right]+\frac{2}{3}\cdot n\lambda\cdot\frac{d}{m}\cdot\frac{\epsilon^{\epsilon/(2m)}+1}{e^{\epsilon/(2m)}-1}}\right). \tag{C. 16}$$

Then, we can get:

$$\Pr\left[\left|\overline{h}(x^*_{i,j})-h(x_{i,j})\right|\ge\lambda\right] \le 2\cdot\exp\left(-\frac{(n\lambda)^2}{O(\frac{dm}{\epsilon})+\lambda\cdot O(\frac{d}{\epsilon})}\right). \tag{C. 17}$$

By the union bound, we have the following inequality:

$$\begin{aligned}\Pr\left[\max_{j\in\{1,\ldots,d\}}\left|\overline{h}(x^*_{i,j})-h(x_{i,j})\right|\ge\lambda\right] &= \bigcup_{j=1}^{d}\Pr\left[\left|\overline{h}(x^*_{i,j})-h(x_{i,j})\right|\ge\lambda\right] \\ &\le \sum_{j=1}^{d}\Pr\left[\left|\overline{h}(x^*_{i,j})-h(x_{i,j})\right|\ge\lambda\right] \\ &= 2d\cdot\exp\left\{-\frac{n\lambda^2}{O\left(\frac{dm}{\epsilon^2}\right)+\lambda O\left(\frac{d}{\epsilon}\right)}\right\}\end{aligned} \tag{C. 18}$$

Then, we set:

$$\delta = 2d\cdot\exp\left\{-\frac{n\lambda^2}{O\left(\frac{dm}{\epsilon^2}\right)+\lambda O\left(\frac{d}{\epsilon}\right)}\right\} \tag{C. 19}$$

To sure that $\max_{j\in\{1,\ldots,d\}}\left|\overline{h}(x^*_{i,j})-h(x_{i,j})\right|<\lambda$ holds with at least $1-\delta$ probability, there exists

$$\lambda = O\left(\frac{\sqrt{d\log(d/\delta)}}{\epsilon\sqrt{|\mathcal{N}(v_i)|}}\right) \tag{C. 20}$$

D. Datasets

For a comprehensive assessment, we evaluated LGA-PGNN on eight benchmark datasets. Cora,[1] Citeseer,[2] and Pubmed[3] are three well-known citation network datasets for node classification task, where nodes and edges represent papers (or publications), and citation relationships, respectively. The labels denote paper category. Moreover, these three datasets are also used to evaluate the effectiveness of link stealing attacks. Deezer[4] offers music streaming to users in three European countries (including Hungary, Croatia, and Romania) and collects a list of their favorite genres. Then, three user friendship network datasets are established, named Deezer HU, Deezer CR, and Deezer RO, respectively. Each node has 84 different genres as node attributes. We label whether users like "R&B" genre as class labels. Specifically, in our attribute inference attack, we treat the "Alternative" genre as a private attribute of the user. We then train a attack model in order to

[1] https://linqs-data.soe.ucsc.edu/public/lbc/cora.tgz
[2] https://linqs-data.soe.ucsc.edu/public/lbc/citeseer.tgz
[3] https://linqs-data.soe.ucsc.edu/public/Pubmed-Diabetes.tgz
[4] http://snap.stanford.edu/data/

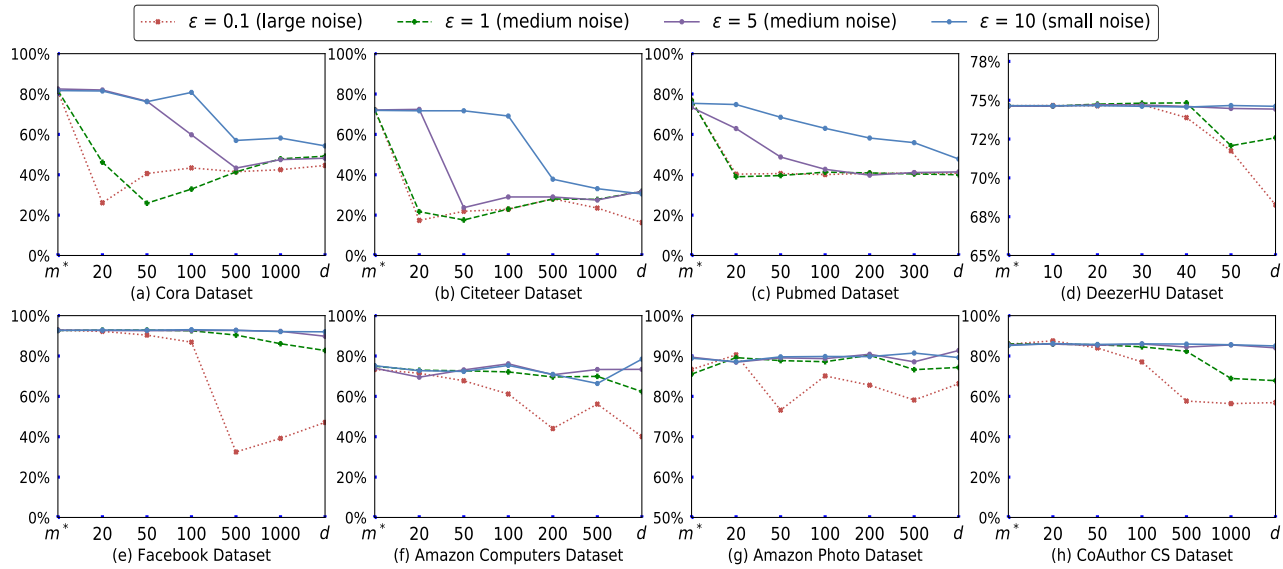


Fig. 8. The effect of perturbation dimension on experimental results.

TABLE VII
MAIN DATASETS USED IN OUR EVALUATION STUDIES

Datasets	# $\lvert\mathcal{V}\rvert$	# $\lvert\mathcal{E}\rvert$	# features	# Classes.
Cora	2485	5069	1433	7
Citeseer	2110	3668	3703	6
Pubmed	19,717	44,324	500	3
DeezerHU	54,573	552,775	83	2
Facebook	22,470	171,002	4714	4
Amazon Computers	13,381	245,778	767	10
Amazon Photo	7487	119,043	745	8
CoAuthor CS	18,333	81,894	6805	15

infer this private attribute of other users. Amazon Computers[5] and Amazon Photo[5] are generated from Amazon's co-purchase graph, where nodes are goods, that are connected by an edge if they are frequently purchased together. The task is to predict the product category. CoAuthor CS[5] is a co-authorship graph generated from the Microsoft Academic Graph dataset, where nodes are authors, and the edge indicates that two authors co-authored a paper. Facebook[4] is a page-page dataset generated by Facebook, where nodes are Facebook pages, and two nodes that like each other are connected by an edge. Table VII provides details of these datasets.

E. Evaluation Metrics

In the following experiments, we evaluate the performance of our privacy-preserving model based on some common performance evaluation metrics in machine learning, including Accuracy, Precision (denoted as P), Recall (denoted as R), and F-score. Precision score is calculated as the ratio of the number of correct positively labeled examples in all positively labeled examples. Recall is defined as the ratio of the number of correct positively labeled examples in all the output examples that should have been labeled positive. F-score is the harmonic mean of precision and recall. The goal of any network intrusion research is to achieve a high value for F-score.

[5]https://github.com/shchur/gnn-benchmark#datasets

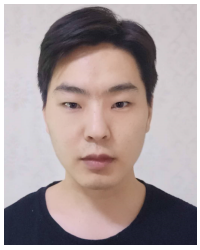
F. Effects of Feature Dimensions

The selection of perturbation dimension has an essential influence on the results. To validity of the perturbation dimension, we compare the classification performance of the baseline model GCN with different perturbation dimensions under the varying privacy budgets. Fig. 8 shows performance trends of the baseline model GCN. Moreover, the changing trend in Fig. 8 shows that when the perturbation dimension m is large, many features are disturbed, resulting in a significant decline in the performance of GCN. In contrast, when m is small, it even performs equally well with the no-privacy case. It is worth mentioning that, as discussed in Sections V-A and V-B, when m is small, users are vulnerable to attribute inference attacks and link stealing attacks. Therefore, this requires us to consider the trade-off between privacy and utility.

REFERENCES

[1] C. Chen et al., "Vertically federated graph neural network for privacy-preserving node classification," in *Proc. IJCAI*, Jul. 2022, pp. 1959–1965.
[2] B. Pinkas, T. Schneider, and M. Zohner, "Faster private set intersection based on OT extension," in *Proc. USENIX*, 2014, pp. 797–812.
[3] S. Wang, Y. Zheng, X. Jia, and X. Yi, "PeGraph: A system for privacy-preserving and efficient search over encrypted social graphs," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 3179–3194, 2022.
[4] S. Sajadmanesh and D. Gatica-Perez, "Locally private graph neural networks," in *Proc. ACM SIGSAC*, Nov. 2021, pp. 2130–2145.
[5] J. Zhou, C. Hu, J. Chi, J. Wu, M. Shen, and Q. Xuan, "Behavior-aware account de-anonymization on Ethereum interaction graph," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 3433–3448, 2022.
[6] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. SSP*, May 2017, pp. 3–18.
[7] X. He, R. Wen, Y. Wu, M. Backes, Y. Shen, and Y. Zhang, "Node-level membership inference attacks against graph neural networks," *CoRR*, vol. abs/2102.05429, pp. 1–15, Feb. 2021.
[8] I. E. Olatunji, W. Nejdl, and M. Khosla, "Membership inference attack on graph neural networks," in *Proc. 3rd IEEE Int. Conf. Trust, Privacy Secur. Intell. Syst. Appl. (TPS-ISA)*, Dec. 2021, pp. 11–20.
[9] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. ACM SIGSAC*, Oct. 2015, pp. 1322–1333.
[10] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *Proc. USENIX*, 2016, pp. 601–618.

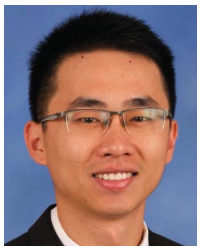
[11] M. Abadi et al., "Deep learning with differential privacy," in *Proc. ACM SIGSAC*, 2016, pp. 308–318.
[12] J. Lee and D. Kifer, "Concentrated differentially private gradient descent with adaptive per-iteration privacy budget," in *Proc. ACM SIGKDD*, Jul. 2018, pp. 1656–1665.
[13] H. Hu, L. Cheng, J. P. Vap, and M. Borowczak, "Learning privacy-preserving graph convolutional network with partially observed sensitive attributes," in *Proc. WWW*, Apr. 2022, pp. 3552–3561.
[14] T. Igamberdiev and I. Habernal, "Privacy-preserving graph convolutional networks for text classification," in *Proc. LREC*, 2022, pp. 338–350.
[15] X. Pei, X. Deng, S. Tian, L. Zhang, and K. Xue, "A knowledge transfer-based semi-supervised federated learning for IoT malware detection," *IEEE Trans. Depend. Sec. Comput.*, vol. 20, no. 3, pp. 2127–2143, May/Jun. 2023.
[16] W. Lin, B. Li, and C. Wang, "Towards private learning on decentralized graphs with local differential privacy," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 2936–2946, 2022.
[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. ICLR*, 2016, pp. 1–14.
[18] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liù, and Y. Bengio, "Graph attention networks," in *Proc. ICLR*, 2018, pp. 12.
[19] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. NIPS*, 2016, pp. 3837–3845.
[20] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. NIPS*, vol. 30, 2017, pp. 1024–1034.
[21] F. Wu, A. H. D. Souza Jr., T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proc. ICML*, vol. 97, 2019, pp. 6861–6871.
[22] J. Du, S. Zhang, G. Wu, J. M. F. Moura, and S. Kar, "Topology adaptive graph convolutional networks," *CoRR*, vol. abs/1710.10370, pp. –13, Oct. 2017.
[23] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proc. ACM SIGSAC*, Oct. 2017, pp. 587–601.
[24] N. Z. Gong and B. Liu, "Attribute inference attacks in online social networks," *ACM Trans. Privacy Secur.*, vol. 21, no. 1, pp. 1–30, Feb. 2018.
[25] M. Kosinski, D. Stillwell, and T. Graepel, "Private traits and attributes are predictable from digital records of human behavior," *Proc. Nat. Acad. Sci. USA*, vol. 110, no. 15, pp. 5802–5805, Apr. 2013.
[26] B. Jayaraman and D. Evans, "Are attribute inference attacks just imputation?" in *Proc. ACM SIGSAC CCS*, Nov. 2022, pp. 1569–1582.
[27] X. He, J. Jia, M. Backes, N. Z. Gong, and Y. Zhang, "Stealing links from graph neural networks," in *Proc. USENIX Secur. Symp.* Berkeley, CA, USA: USENIX Association, 2021, pp. 2669–2686.
[28] F. Wu, Y. Long, C. Zhang, and B. Li, "LINKTELLER: Recovering private edges from graph neural networks via influence analysis," in *Proc. IEEE SSP*, May 2022, pp. 2005–2024.
[29] A. Kolluri, T. Baluta, B. Hooi, and P. Saxena, "LPGNet: Link private graph networks for node classification," in *Proc. ACM SIGSAC CCS*, Nov. 2022, pp. 1813–1827.
[30] S. Hidano and T. Murakami, "Degree-preserving randomized response for graph neural networks under local differential privacy," *CoRR*, vol. abs/2202.10209, pp. 1–13, Feb. 2022.
[31] K. Tran et al., "Heterogeneous randomized response for differential privacy in graph neural networks," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2022, pp. 1582–1587.
[32] Y. Rong, W. Huang, T. Xu, and J. Huang, "DropEdge: Towards deep graph convolutional networks on node classification," in *Proc. ICLR*, 2020, pp. 1–18.
[33] T. Zhao, Y. Liu, L. Neves, O. J. Woodford, M. Jiang, and N. Shah, "Data augmentation for graph neural networks," *CoRR*, vol. abs/2006.06830, pp. 1–15, Jun. 2020.
[34] S. Liu et al., "Local augmentation for graph neural networks," in *Proc. ICML*, vol. 162, 2022, pp. 14054–14072.
[35] N. Wang et al., "Collecting and analyzing multidimensional data with local differential privacy," in *Proc. ICDE*, Apr. 2019, pp. 638–649.
[36] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014.
[37] 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, pp. 1–20, Jun. 2023.
[38] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. ICLR*, Feb. 2019, pp. 1–17.
[39] N. Phan, X. Wu, H. Hu, and D. Dou, "Adaptive Laplace mechanism: Differential privacy preservation in deep learning," in *Proc. ICDM*, Nov. 2017, pp. 385–394.



Xinjun Pei (Student Member, IEEE) is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Central South University, Changsha, China. Since 2017, he has been engaged in the direction of information security. His research interests include graph representation learning, distributed computing systems, and cyberspace security.

Xiaoheng Deng (Senior Member, IEEE) received the Ph.D. degree in computer science from Central South University, Changsha, Hunan, China, in 2005. Since 2006, he has been an Associate Professor and then a Full Professor with the School of Computer Science and Engineering, Central South University. His research interests include wireless communications and networking, edge computing, congestion control for wired/wireless networks, cross-layer route design for wireless mesh networks and ad hoc networks, online social network analysis, and distributed computing systems. He is a Senior Member of CCF and a member of the CCF Pervasive Computing Council. He was the Chair of CCF YOCSEF CHANG SHA from 2009 to 2010.

Shengwei Tian received the B.S., M.S., and Ph.D. degrees from the School of Information Science and Engineering, Xinjiang University, Ürümqi, China, in 1997, 2004, and 2010, respectively. Since 2002, he has been a Teacher with the School of Software, Xinjiang University, where he is currently a Professor. His research interests include artificial intelligence, natural language processing, and cyberspace security.



Jianqing Liu (Member, IEEE) received the B.Eng. degree from the University of Electronic Science and Technology of China in 2013 and the Ph.D. degree from the University of Florida in 2018. He is currently an Assistant Professor with the Department of Computer Science, NC State University, Raleigh, NC, USA. His research interests include wireless communications and networking, security, and privacy. He received the U.S. National Science Foundation Career Award in 2021. He was a recipient of several best paper awards, including the 2018 Best Journal Paper Award from the IEEE Technical Committee on Green Communications and Computing (TCGCC).

Kaiping Xue (Senior Member, IEEE) received the bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003, and 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 Post-Doctoral Researcher with the Department of Electrical and Computer Engineering, University of Florida. Currently, he is a Professor with the School of Cyber Science and Technology, USTC. His research interests include next-generation internet architecture design, transmission optimization, and network security. He is an IET Fellow. He serves on the editorial board for several journals, including IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, and IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT. He has also served as a (Lead) Guest Editor for many reputed journals/magazines, including IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, *IEEE Communications Magazine*, and *IEEE Network*.