# Towards Private Learning on Decentralized Graphs With Local Differential Privacy

Wanyu Lin, *Member, IEEE*, Baochun Li, *Fellow, IEEE*, and Cong Wang, *Fellow, IEEE*

***Abstract*— Many real-world networks are inherently decentralized. For example, in social networks, each user maintains a local view of a social graph, such as a list of friends and her profile. It is typical to collect these local views of social graphs and conduct graph learning tasks. However, learning over graphs can raise privacy concerns as these local views often contain sensitive information. In this paper, we seek to ensure private graph learning on a decentralized network graph. Towards this objective, we propose *Solitude*, a new privacy-preserving learning framework based on graph neural networks (GNNs), with formal privacy guarantees based on edge local differential privacy. The crux of *Solitude* is a set of new delicate mechanisms that can calibrate the introduced noise in the decentralized graph collected from the users. The principle behind the calibration is the intrinsic properties shared by many real-world graphs, such as sparsity. Unlike existing work on locally private GNNs, our new framework can simultaneously protect node feature privacy and edge privacy, and can seamlessly incorporate with any GNN with privacy-utility guarantees. Extensive experiments on benchmarking datasets show that *Solitude* can retain the generalization capability of the learned GNN while preserving the users' data privacy under given privacy budgets.**

***Index Terms*— Privacy-preserving graph learning, graph neural networks, differential privacy, decentralized network graph.**

## I. Introduction

MANY problems in scientific domains, ranging from computer networks [1] and social networks to biomedicine and healthcare [2], [3], can be naturally cast as problems of property learning on graphs. Typically, these graph domains contain sensitive information. In computer networks, for example, the goal of botnets detection is to isolate the botnet nodes, where the problem can be formulated as binary node classification task on massive background Internet communication graphs [4]. However, the Internet Service Providers (ISPs) may be reluctant to share traffic observations (modeled as edges in the communication graphs [4]). Likewise, in social networks, users' contact lists, profile information, likes or comments, etc., should be kept be private, as most users are not willing to release their contact lists to strangers. Inevitably, the use of sensitive and private graph data requires principled and rigorous privacy guarantees. On the other hand, among various graph learning algorithms, graph neural networks (GNNs) have exhibited superior performance [4], due to their efficiency and inductive learning capability [5]. A recent work shows that the edge information of graphs can be recovered from GNNs through influence analysis [6]. Therefore, it is appealing to tailor the GNNs to perform graph learning tasks while still preserving user data privacy. A plausible approach is local differential privacy (LDP) [7], where each user locally obfuscates their share of data before sending them to a data curator (who may be malicious). As the data curator performs model learning over the obfuscated data, data privacy may be preserved under given privacy budgets. However, most existing LDP techniques for graphs mainly focused on graph statistics analysis while protecting the privacy of edge/link information. Typical graph statistics include subgraph counting [8] (e.g., triangles and $k$-stars counting) and graph metric estimations [9] (e.g., clustering coefficient, modularity, or centrality estimation), which are not designed for GNNs.

Learning over the obfuscated graphs with GNNs is quite challenging. In general, training deep learning models with strong differential privacy guarantees comes at a significant cost in utility [10], [11]. Specifically, in the context of graph learning, graph data usually contains *node feature* information and *graph structure* information. The combinatorial nature of graph structures makes the private learning problem more complex than other domains. These can be explained as follows. In GNNs, the node representations — can be used for various downstream tasks — are learned in a way that node information is aggregated and propagated via links through the message passing framework during training [5], [12]. In node classification tasks or link prediction tasks, the training samples (nodes or links in the graph) of the learning model are interdependent. In contrast, the existing work on LDP for tabular data assumes that each user's data is independently and identically drawn from an underlying distribution [13], [14], which is problematic in the context of graphs.

While private graph learning with GNNs is still a nascent research topic, a recent proposal attempts to preserve the privacy of the node features [15]. However, this model raises

Manuscript received 17 January 2022; revised 11 April 2022 and 27 June 2022; accepted 12 July 2022. Date of publication 11 August 2022; date of current version 18 August 2022. The work of Wanyu Lin was supported by the Research Fund at The Hong Kong Polytechnic University under Grant P0035763 and Grant P0042687. The work of Cong Wang was supported by the Research Grants Council of Hong Kong under Grant CityU 11217620, Grant N_CityU139/21, and Grant RFS2122-1S04. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Lalitha Sankar. *(Corresponding author: Wanyu Lin.)*

Wanyu Lin is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: wanylin@comp.polyu.edu.hk).

Baochun Li is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 1A1, Canada (e-mail: bli@ece.toronto.edu).

Cong Wang is with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: congwang@cityu.edu.hk).

Digital Object Identifier 10.1109/TIFS.2022.3198283

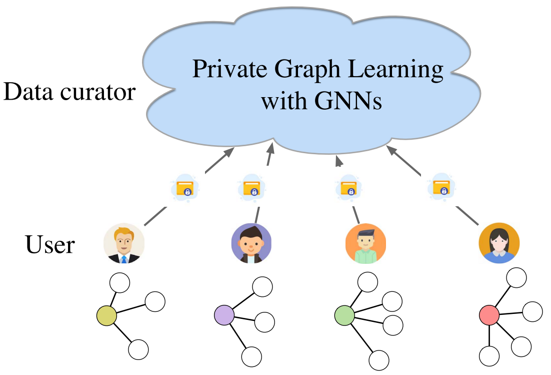


Fig. 1. Illustration of private learning over decentralized graph network: 1) each user holds a data share including a local neighbour list and the profile information; 2) the data curator collects the obfuscated data and conducts learning over the collected noisy graphs.

several privacy and security issues as it assumes that the data curator holds the global graph structure. If the topological features contain sensitive information, this approach may incur information leakage as the data curator can directly access the global topology for the message passing process. Therefore, in this paper, we study the problem of differentially private graph learning with GNNs on a decentralized network graph, as shown in Fig. 1. In particular, the data curator cannot directly access the global structure of the graph. In other words, both the node feature and graph structure information should be protected against the data curator.

Privacy-preserving learning under the setting of decentralized network graphs has various applications, including but not limited to social network analysis and mobile computing. In principle, some social networks are inherently decentralized and distributed, such as Synereo [16]. Though in some other social networks, e.g., Facebook, there exists a centralized party that holds the knowledge of the global network, that party may choose not to share it with a third-party (corresponding to the data curator) for analyzing, due to legal issues or other business concerns.

With the prevalence of decentralized network graphs, we propose a new framework consisting of a set of mechanisms, called *Solitude*, to tailor the GNNs for decentralized graph analysis under local differential privacy. Our framework has provable privacy guarantees based on local differential privacy. Specifically, we leverage the notion of edge local differential privacy proposed in [17]. To protect the privacy of neighbor lists, each user applies Warner's randomized response mechanism [18] to obfuscate their neighbor lists before sending them to the data curator. For protecting node/user[1] feature privacy, we further incorporate a multi-bit mechanism for multi-dimensional feature perturbation [15], [19], which can ensure that the high-dimensional feature vector of every user can be protected. However, learning over obfuscated graphs introduces challenges, as it could significantly degrade the generalization capability of the GNN. Precisely, the learned GNN may overfit the noisy graphs and generalize poorly to unseen nodes.

To this end, we propose new mechanisms for graph structure calibration and feature vector calibration, respectively. Essentially, obfuscating graph structure via a randomized response mechanism introduces edge deletion, addition, and rewiring. Though the obfuscating process satisfies our privacy goal, we theoretically and empirically analyze that this process tends to return a much denser graph than the original one, violating the sparsity property — an inherent property exhibited in many real-world graphs [20]. Therefore, we propose encouraging the sparseness of the graph structure during training as a calibration step to reduce the effect of the noise. In addition, a node might be linked to nodes with task-specific "noise," leading to the aggregation of non-smooth features in GNNs. Inspired by the assumption of *feature smoothness* in GNNs, we leverage a feature smoothing component before training to reduce the effect of the feature noise. To further boost the prediction accuracy, a label smoothness component is adopted, which is inspired by the principles of label propagation algorithms [21], [22].

In a nutshell, our original contributions are listed as follows. We propose a new privacy-preserving learning framework for decentralized network graphs based on graph neural networks. It consists of a set of mechanisms that can provide local differential privacy for every user yet maintaining the generalization capability of the learned GNN. We formally analyze that our mechanisms preserve local differential privacy for every user, particularly on the notion of edge differential privacy. Towards our goal, we theoretically and empirically analyze the overfitting problem while learning over the noisy graphs. Different from the literature on locally private GNNs, *Solitude* can preserve edge privacy and node feature privacy for every user simultaneously. Our extensive array of experiments on benchmarking datasets demonstrated that *Solitude* can significantly improve the privacy-utility guarantees on canonical graph learning benchmarks. We also empirically show that our framework can seamlessly integrate with any GNN architectures, such as GCN [12] and GraphSage [5], with privacy-utility guarantees.

## II. Problem Setup

### A. Notations and Problem Definition

*1) Notations:* We consider a network graph, denoted as $\mathcal{G} = (\mathcal{V}, \mathbf{A}, \mathbf{X})$. Specifically, we consider the decentralized setting, where the entire graph is decentralized over users/nodes $\mathcal{V} = \{v_1, \cdots, v_{|\mathcal{V}|}\}$. Precisely, each user $v_i$ holds locally a neighbor list depicted as $\mathbf{a}_i$, which can be modeled as an $|\mathcal{V}|$-binary vector, and a node attribute/feature vector $\mathbf{x}_i \in \mathbb{R}^{|D|}$, where $|D|$ is the dimension of the user/node feature vector. The corresponding adjacency matrix of the entire graph can be represented as $\mathbf{A} = \{\mathbf{a}_1, \cdots, \mathbf{a}_{|\mathcal{V}|}\}$, where $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, and the node attribute matrix $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_{|\mathcal{V}|}\}$, where $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times|D|}$. In the context of social network graphs, for example, each user locally holds their friend list and the profile information (e.g., WeChat [23]). Without loss of generality, we assume $\mathcal{G}$ is a directed graph; this reflects

[1] In many applications, such as social networks, each node represents a user; we use "node" and "user" interchangeably.

TABLE I
NOTATIONS

| Notation | Descriptions |
|---|---|
| $\mathbf{a}_i$ | the adjacency list of user i |
| $\tilde{\mathbf{a}}_i$ | randomized adjacency list of user i |
| $\mathbf{x}_i$ | the feature vector of user i |
| $\hat{\mathbf{x}}_i$ | the encoded feature vector of user i |
| $\tilde{\mathbf{x}}_i$ | the rectified feature vector of user i |
| $\mathcal{M}_a$ | the randomized mechanism for adjacency lists |
| $\mathcal{M}_x$ | the randomized mechanism for node features |
| $\epsilon_x$, $\epsilon_a$ | privacy budgets for features and adjacency lists |
| $\lvert\mathcal{N}(\cdot)\rvert$ | the node degree of user i |
| $\mathbf{A}^c$ | the adjacency matrix after calibration |
| $\mathbf{X}^c$ | the node feature matrix after calibration |
| $\lvert\lvert \cdot \rvert\rvert_{\mathbf{F}}^2$ | the Frobenius norm of a matrix |
| $\lvert\lvert \cdot \rvert\rvert_1$ | $l_1$ norm operator |
| $\boldsymbol{\theta}$ | the model parameters |
| $\lambda_1$, $\lambda_2$ | the regularization coefficients |

the real-world application domains, e.g., the follower-followee relationships.

Consistent with prior work [15], we focus on the node classification task with graph neural networks. Specifically, we follow the standard node classification setting, which is commonly employed in various literature [5], [12]. Given a set of labeled nodes $\mathcal{V}_l \subset \mathcal{V}$, with class labels from $\mathcal{Y} = \{y_1, y_2, y_3, \cdots, y_K\}$ and a set of unlabeled nodes $\mathcal{V}_u \subset \mathcal{V}/\mathcal{V}_l$, the goal of node classification is to map each node $v \in \mathcal{V}$ to one class in $\mathcal{Y}$.

*2) Problem Definition:* We assume that the data curator is an untrusted party, and it can access the set/index of nodes/users $\mathcal{V} = \{v_1, \cdots, v_{|\mathcal{V}|}\}$ and the node labels of the training set $\mathcal{V}_l$. However, the data curator could not directly access the feature matrices $\mathbf{X}$ and $\mathbf{A}$, which are decentralized among the users and private to the users. The data curator is allowed to collect information from the users, and performs node classification with GNNs over the noisy graphs.

Therefore, our ultimate goal is to obtain a set of mechanisms that 1) the data curator can collect data portions from each user while preserving the user data privacy; 2) the data curator can train a GNN for node classification over the collected noisy graph with the best possible generalization capability. Without loss of generality, model generalization capability is measured by the prediction accuracy on the held-out test set that has not been seen during training. Note that, different from the state-of-the-art on locally private GNN [15] — LPGNN, we consider a more advanced setting that the global topology of the graph is not accessible to the data curator. Concretely, user $v_i$'s private data includes the neighbor list, or called *adjacency list*, $\mathbf{a}_i$ and the node attribute/feature vector $\mathbf{x}_i$. In what follows, we first briefly introduce some necessary background on graph learning with GNNs and the notion of local differential privacy on graphs to facilitate a better understanding of our solution. The mathematical notations used in this paper are summarized in Table I.

### B. Message Passing Graph Neural Networks

Graph neural networks (GNNs) are tailored to learn and model information structured as graph data. It is a family of graph message passing architectures that incorporate graph structure and node feature vectors to learn a dense representation of a node or the entire graph. In principle, GNNs share a neighborhood aggregation strategy, where the node representations are refined via iteratively aggregating the representations from its neighboring nodes in the graph. Representative GNNs are graph convolutional networks, which use mean pooling for aggregation [12], and GraphSage that aggregates the node features via mean/max/LSTM pooling [5].

Taking GCNs as an example, the basic operator for the neighborhood information aggregation is the element-wise **mean**. After $L$ iterations of aggregation, a node's representation can capture the structural information within its $L$-hop graph neighborhood, which can be formulated as:

$$h_v^{l+1} \leftarrow \sigma\left(W^l \cdot \textbf{mean}\left(\{h_v^l\} \cup \{h_u^l, \forall u \in \mathcal{N}(v)\}\right)\right) \quad (1)$$

where $W^l$ is a trainable matrix for layer $l$, $\sigma$ denotes a nonlinear activation function. Note that, except for **mean** operator, there are many other operators for neighborhood information aggregation, such as **max** pooling. Due to their superior performance, these operators have been the core of many graph neural networks. For more details on other GNN variations, we refer the interested readers to the existing survey [24].

### C. Local Differential Privacy

Local differential privacy (LDP) has emerged as the *de facto* solution for collecting private data and performing statistical queries, such as mean, counting, etc. In principle, it is a privacy metric to protect the sensitive information of individuals from the data curator [7], [25]. In the setting of LDP, each user does not trust the data curator. Before sending her share of data to the data curator, each user locally perturbs the data portion with a differentially private mechanism. The perturbed data is not meaningful individually but can be used for data analytics when aggregated.

In the context of graph data, a differentially private mechanism can be designed for edge differential privacy [26], or node differential privacy [27]. In essence, edge differential privacy ensures that a randomized mechanism does not reveal the addition or deletion of an edge in the neighbor list of an individual. In contrast, a randomized mechanism for node differential privacy hides the deletion or addition of a node along with its link list. Consistent with prior works [9], [17], we adopt the notion of edge local differential privacy (LDP) based on a user's neighbor list.

Formally, let $\mathbf{a}_i = (a_{i,1}, \cdots, a_{i,|\mathcal{V}|}) \in \{0, 1\}^{|\mathcal{V}|}$ be the neighbor list of the user $v_i$, where $\mathbf{a}_i$ is the $i$-th row of the adjacency matrix $\mathbf{A}$ of the global network graph $\mathcal{G}$. Stated differently, the adjacency matrix of $\mathcal{G}$ can be represented as $\mathbf{A} = \{\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_{|\mathcal{V}|}\}$. Then edge LDP can be defined as follows [8], [17]:

*Definition 1 (Edge Local Differential Privacy):* A randomized mechanism $\mathcal{M}$ satisfies $\epsilon$-edge local differential privacy ($\epsilon$-edge LDP) if and only if for any two neighbor lists $\mathbf{a}$ and $\tilde{\mathbf{a}}$, such that $\mathbf{a}$ and $\tilde{\mathbf{a}}$ only differ in one bit, and any

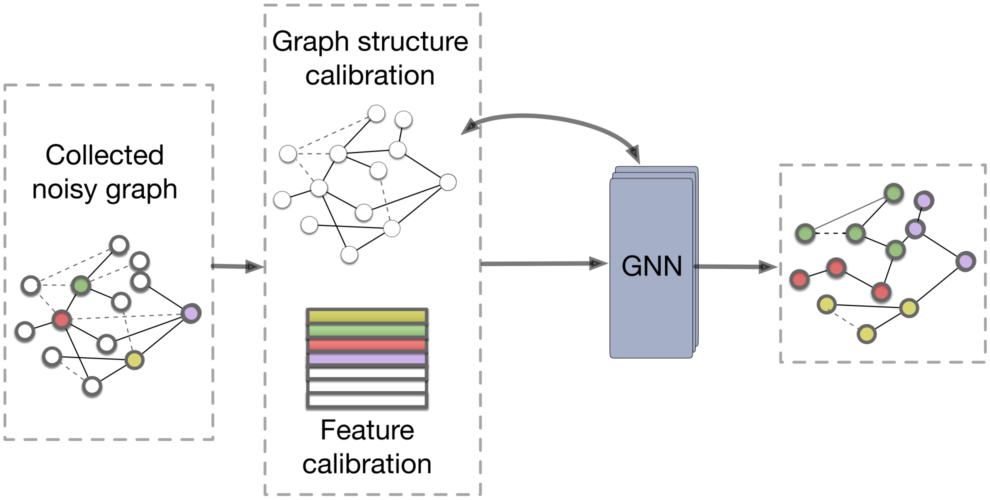


Fig. 2. Illustration of the private GNN training at the data curator side. It consists of two components: 1) a feature denoising component to reduce the effect of non-smooth aggregation; it is carried out before the training process; 2) a graph structure denoising component to enforce the graph structure's sparseness; this process is jointly optimized with the model training.

$s \subseteq \mathbf{range}(\mathcal{M})$, we have

$$\frac{\mathbf{Pr}[\mathcal{M}(\mathbf{a}) = s]}{\mathbf{Pr}[\mathcal{M}(\tilde{\mathbf{a}}) = s]} \leq e^{\epsilon}, \tag{2}$$

where $\epsilon$ is the privacy budget.

Note that, $\epsilon$-edge LDP in Definition 1 protects one single bit in a neighbor list with privacy budget $\epsilon$. By adopting the notion of group privacy [18], $\epsilon$-edge LDP can be used to protect $k \in \mathbb{N}$ edges. Concretely, if $\mathcal{M}$ provides $\epsilon$-edge LDP, then for any two neighbor lists $\mathbf{a}$ and $\tilde{\mathbf{a}}$ that differ in $k$ edges and any $s \subseteq \mathbf{range}(\mathcal{M})$, we have

$$\frac{\mathbf{Pr}[\mathcal{M}(\mathbf{a}) = s]}{\mathbf{Pr}[\mathcal{M}(\tilde{\mathbf{a}}) = s]} \leq e^{k\epsilon}, \tag{3}$$

where $k$ edges are protected with privacy budget $k\epsilon$.

*Properties.* LDP satisfies composition property and transformation invariance [27]. Specifically, composition property enables the modular design of mechanisms: if all the components of a mechanism are differentially private, so is their composition. The transformation invariance demonstrates that performing post-processing on the output of an algorithm that satisfies LDP does not affect the privacy guarantee.

## III. *Solitude*: Our Proposed Framework

This section describes the main components of our proposed framework, called *Solitude*, toward differentially private training of GNN over private graph data, including the node feature vectors and the neighbor lists that are decentralized across all users. Specifically, there are two stages in our framework. In the first stage, the data curator sends a query to each user $v_i$ once, and then each user $v_i$ independently sends an answer – the obfuscated data share, including the obfuscated version of node feature vector $\tilde{\mathbf{x}}_i$ and the obfuscated adjacency list $\tilde{\mathbf{a}}_i$ (shown in Fig. 1). In the second stage, the data curator performs training of the GNN over the noisy graph composed of the obfuscated data shares from all users. The second stage of our *Solitude* is illustrated in Fig. 2.

In what follows, we first introduce the technical details of the randomized mechanisms used for preserving data privacy satisfying local differential privacy. Then we theoretically and empirically analyze the overfitting issue caused by the randomized mechanisms. We show how to calibrate the introduced noise such that the generalization capability of the trained GNN can be retained.

### A. Obfuscating Local Data Under Local Differential Privacy

In the setting of the decentralized network graph, each user $v_i$ holds a data portion of the entire graph, including their feature vector $\mathbf{x}_i$ and the adjacency list $\mathbf{a}_i$, both of which are private to the user. In the subsequent paragraphs, we describe the used mechanisms for protecting the privacy of the feature vector and the adjacency list, respectively.

*1) Randomized Adjacency List:* Intrinsically, an adjacency list is a binary bit vector, denoted as $\mathbf{a}_i = \{a_{i,1}, \cdots, a_{i,|\mathcal{V}|}\}$, in which $a_{i,j} = 1$ indicates the link between $v_i$ and $v_j$. Without loss of generality, we leverage of a common methodology, called *randomized response* [17], [18], to impel local differential privacy. Specifically, each user flips each bit of her adjacency list with a probability $p$ constrained by a given privacy budget. More formally, given a privacy budget $\epsilon_a$, the randomized adjacency list, denoted as $\tilde{\mathbf{a}}_i = \{\tilde{a}_{i,1}, \cdots, \tilde{a}_{i,|\mathcal{V}|}\}$, is obtained as follows:

$$\tilde{a}_{i,j} = \begin{cases} a_{i,j}, & q = \frac{\mathbf{e}^{\epsilon_a}}{1+\mathbf{e}^{\epsilon_a}} \\ 1 - a_{i,j}, & p = \frac{1}{1+\mathbf{e}^{\epsilon_a}} \end{cases}, \tag{4}$$

where $q = 1 - p$ is the probability of retaining a particular bit in the adjacency list.

*Theorem 1:* The randomized adjacency list mechanism $\mathcal{M}_a$ satisfies $\epsilon_a$-edge local differential privacy.

*Proof:* Let us consider the case that $\mathbf{a}_i$ and $\tilde{\mathbf{a}}_i$ only differ in one bit. Concretely, we assume that $a_{i,j} \neq \tilde{a}_{i,j}$, and given any output $s = (s_1, \cdots, s_n)$ from $\mathcal{M}$, we have

$$\frac{\mathbf{Pr}[\mathcal{M}_a(\mathbf{a}_i) = s]}{\mathbf{Pr}[\mathcal{M}_a(\tilde{\mathbf{a}}_i) = s]} = \frac{\mathbf{pr}[a_{i,1} \to s_1)] \cdots \mathbf{pr}[a_{i,n} \to s_n)]}{\mathbf{pr}[\tilde{a}_{i,1} \to s_1)] \cdots \mathbf{pr}[\tilde{a}_{i,n} \to s_n)]} \tag{5}$$

$$= \frac{\mathbf{pr}[a_{i,j} \to s_j)]}{\mathbf{pr}[\tilde{a}_{i,j} \to s_j)]} = \frac{q}{p} \leq e^{\epsilon_a}. \tag{6}$$

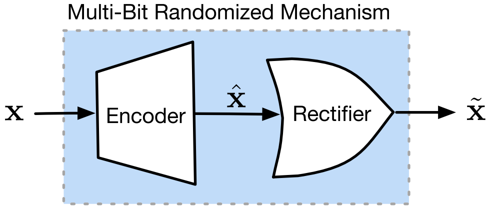


Fig. 3. Illustration of multi-bit mechanism. The encoder encodes the multi-dimensional features $\mathbf{x}_i \in \mathcal{R}^{|D|}$ into $\hat{\mathbf{x}}_i \in \{-1, 0, 1\}^{|D|}$, with a probability defined by the privacy budget. The rectifier is to calibrate the encoded vector $\hat{\mathbf{x}}_i$ to ensure the outcome $\tilde{\mathbf{x}}_i$ is statistically unbiased.

*2) Randomized Feature Vector:* If the privacy of the node feature is also the concern, we further leverage a multi-bit mechanism to protect its privacy for every user. The randomized mechanism for feature vector in our framework follows the similar outline as the mechanism used in [15] and [19]. Specifically, the multi-bit mechanism has two components: an encoder and a rectifier, as shown in Fig. 3. To randomize a feature vector $\mathbf{x}_i \in \mathcal{R}^{|D|}$, where each element $x_{i,j}$ falls into the range $[x_{\mathbf{min}}, x_{\mathbf{max}}]$, the encoder first uniformly samples $m$ features out of the $D$ dimensions. Each of the selected features is encoded into $-1$ or 1, with a probability formulated as

$$\frac{1}{\mathbf{e}^{\epsilon_x/m}+1} + \frac{x_{i,j}-x_{\mathbf{min}}}{x_{\mathbf{max}}-x_{\mathbf{min}}} \cdot \frac{\mathbf{e}^{\epsilon_x/m}-1}{\mathbf{e}^{\epsilon_x/m}+1}. \qquad (7)$$

Correspondingly, the rest of the $d - m$ features are mapped to 0s. The rectifier is to calibrate the encoded vector $\hat{\mathbf{x}}$ to ensure the outcome of the randomized mechanism $\tilde{\mathbf{x}}$ is statistically unbiased. Formally, the rectifier is instantiated as

$$\begin{aligned} \mathbf{Rec}(\hat{x}_{i,j}) = \frac{|D| \cdot (x_{\mathbf{max}} - x_{\mathbf{min}})}{2m} \cdot \frac{\mathbf{e}^{\epsilon_x/m}+1}{\mathbf{e}^{\epsilon_x/m}-1} \cdot \hat{x}_{i,j} & \qquad (8) \\ + \frac{x_{\mathbf{max}} + x_{\mathbf{min}}}{2}. & \qquad (9) \end{aligned}$$

*Theorem 2:* The randomized mechanism for node feature vector $\mathcal{M}_x$ preserves $\epsilon_x$ differential privacy of every user.

Due to the page limitation, we refer to the detailed proof of Theorem 2 to [15], [19]. Note that the multi-bit mechanism for feature vector is different from the notion of node local differential privacy (node-LDP). Specifically, node-LDP hides the deletion or addition of a node along with its neighbor list; it is out of the scope in this paper. With the composition property of local differential privacy as described in Section II-C, we arrive at the following corollary:

*Corollary 1:* The randomized mechanisms ($\mathcal{M}_a$ and $\mathcal{M}_x$) together satisfy $\epsilon_a + \epsilon_x$ differential privacy of every user.

### *B. Private GNN Training With Calibration*

Now with the collected graph data from all users, the data curator can reconstruct the global graph, and it is ready to perform graph analytics with message passing GNNs; the task is instantiated with node classification in this work. Yet it can be generalized to other graph learning tasks, such as link prediction, clustering coefficient prediction [28], as these tasks all share the same graph message passing architecture. Unfortunately, the collected graph is noisy; aggregating and propagating the noisy information leads to over-fitting which degenerates the generalization ability.

TABLE II

CLASSIFICATION ACCURACY (%) ON CORA AND CITESEER, WITH $\epsilon_x = 1$. THE PERFORMANCE ARE INSIGNIFICANT IN VARIOUS EDGE PRIVACY BUDGETS

| $\epsilon_a$ | 7.0 | 7.3 | 7.5 | 7.7 | 7.9 |
|---|---|---|---|---|---|
| CORA | $50.9 \pm 3.8$ | $54.9 \pm 3.7$ | $57.2 \pm 2.5$ | $61.4 \pm 2.6$ | $63.7 \pm 2.4$ |
| CITESEER | $35.0 \pm 1.8$ | $39.1 \pm 1.3$ | $42.3 \pm 1.5$ | $44.5 \pm 1.2$ | $45.7 \pm 1.8$ |

*1) Why Does the Performance Degrade?:* Although the randomized collection of the adjacency lists and feature vectors satisfies our privacy goal, the reconstructed graph does not reflect the original decentralized social graph well. From the perspective of the graph structure, it tends to return a much denser graph than the original one. For example, a citation network Cora [29] has $|\mathcal{V}| = 2708$ nodes, and the average node degree is 3.89. Mathematically, the randomized mechanism $\mathcal{M}_a$ introduces $p \times |\mathcal{V}|^2 = 6681$ edge flipping with $\epsilon_a = 7$ in expectation. We empirically analyze the average node degree of the reconstructed graph in the setting of $\epsilon_a = 7$; it is 6.35 by averaging the results of 5 repeats. Concretely, it introduces 6634 edges that may be task-irrelevant, averaged over 5 repeats. A node might be linked to nodes with task-specific "noisy" edges due to the randomized flipping. Aggregating messages from these nodes would compromise the quality of the node embedding and lead to undesirable predictions in the downstream tasks.

To visualize the performance degradation, we evaluated the task of document classification with different values of privacy budgets on Cora [29] and CiteSeer [30] respectively. As observations are similar in other values of $\epsilon_x$, we present the results of $\epsilon_x = 1$ while changing the value of $\epsilon_a$ in Table II. We observed that under the randomized mechanisms, the performance of GNNs is non-significant. In what follows, we are interested to exploit the intrinsic properties of the data to boost the classification accuracy with further gains when the graph is collected satisfying given privacy goals.

*2) Graph Structure Denoising:* The randomized flipping introduces "noisy" edges that can degrade the generalization performance of the GNNs. These edges tend to connect nodes within different communities or with different labels, and they should be pruned for better learning performance. According to the analysis mentioned above, the randomized flipping increases the density of the graph. Therefore, we propose to calibrate the collected noisy graph by encouraging the sparseness of the graph structure. In particular, we calibrate the graph structure by minimizing the $l_1$ norm of the calibrated adjacency matrix, denoted as $||\mathbf{A}^c||_1$. Therefore, the graph structure calibration process can be formulated as an optimization problem, shown in Eq. 10.

$$\min_{\mathbf{A}^c} ||\tilde{\mathbf{A}} - \mathbf{A}^c||_{\mathbf{F}}^2 + \lambda ||\mathbf{A}^c||_1, \qquad (10)$$

where $\mathbf{A}^c$ represented the adjacency matrix after calibration, and $\lambda$ control the associated calibration level. The first term is to ensure the calibrated matrix to be close to the collected graph topology. Concretely, the distance of the calibrated matrix and the collected matrix is measured by the Frobenius norm. The Frobenius norm of a matrix $\mathbf{A}$ is defined by $||\mathbf{A}||_{\mathbf{F}}^2 = \Sigma a_{i,j}^2$. We are aware of the drawbacks of reusing notations. $\mathbf{A}$ in the definition of the Frobenius norm represents any matrix for simplicity.

*3) Node Feature Vector Denoising:* In GNNs, it computes the new representation of a node by aggregating and propagating information from its neighbors [5]. The learned representations of connected nodes tend to be similar. Stated differently, for message passing graph neural network to work, a certain assumption, called the *smoothness assumption*, has to hold. This reflects the real-world phenomenon on graphs from various domains. For example, two connected users in a social graph are likely to share similar features, fulfilling the property of *feature smoothness*. However, in the noisy graphs, the GNNs may result in degenerated node representations due to the non-smooth features aggregating from the neighbors, leading to performance degradation of the learned GNN.

To address the above issue, we leverage a feature smoothing component that applies a mean aggregator [15] to enhance the node features – denoising via mean aggregation of the node features from the neighbors. Specifically, instead of using the rectified features $\tilde{\mathbf{X}}$, we refine the features of each node by averaging the feature vector from their neighbors within $l$-hop. Formally, the process of feature smoothing within 1-hop can be formulated as:

$$\mathbf{x}_i^c = \sum_{v_j \in \mathcal{N}(v_i)} \frac{\tilde{\mathbf{x}}_j}{|\mathcal{N}(v_i)||\mathcal{N}(v_j)|}. \tag{11}$$

Our feature smoothing component is executed with $l_x$ times, which is data-driven and needs to be tuned to avoid over-smoothing problem.

These denoising processes, including graph-structure denoising and feature vector denoising, are designed as calibration steps for better learning of the GNNs. According to the transformation invariance of LDP [27], these processes would not affect the privacy guarantee. Stated differently, the output of these processes is still noisy and does not reflect the private data portion of each user. Nevertheless, the feature smoothing operation is designed to reduce the noise effect that may degrade the generalization capability of the GNN.

*4) Model Training:* To evaluate the effectiveness of *Solitude* for privacy-preserving graph learning tasks, we instantiate the graph learning task with node classification. We denote the target classifier as $f(\tilde{\mathbf{x}}) = \mathbf{arg}\max_y p(y|\tilde{\mathbf{x}})$, where $p(y|\tilde{\mathbf{x}}) = g(\mathbf{A}^c; \mathbf{X}^c, \boldsymbol{\theta})$. More specifically, $g$ represents the GNN model and $\boldsymbol{\theta}$ are the learned model parameters given the obfuscated graph $\tilde{\mathcal{G}} = (\tilde{\mathbf{A}}, \tilde{\mathbf{X}})$ and the denoising mechanisms. Note that the generalization capability of the learned model is measured by the prediction accuracy over an obfuscated/noisy test set that is not seen during training. To further boost the prediction accuracy, we further incorporate a label smoothing component, which is inspired by the principles of label propagation algorithms [21], [22] — node labels are propagated and aggregated along edges in the graph. Formally, the label smoothing component within 1-hop is formulated as:

$$p(y^c|\tilde{\mathbf{x}}) = \sum_{v_j \in \mathcal{N}(v_i)} \frac{p(y|\tilde{\mathbf{x}})}{|\mathcal{N}(v_i)||\mathcal{N}(v_j)|}. \tag{12}$$

The label smoothing component can be carried out with $l_y$ times, similar to the feature smoothing component. In Section IV, we will show that for node classification tasks, improving privacy-utility guarantees needs more labeled samples for training.

In general, the model parameters are obtained by minimizing the cross-entropy error over all labeled samples:

$$\mathcal{L}_{\mathbf{GNN}}(\mathbf{A}^c, \mathbf{X}^c, \boldsymbol{\theta}) = -\sum_{v_i \in \mathcal{V}_l} \sum_{k=1}^{K} y_{ik} \ln p(y_k^c|\tilde{\mathbf{x}}), \tag{13}$$

where $\mathcal{V}_l$ is the set of node indices that have labels, and K is the number of classes/labels.

Intuitively, the denoising processes can be regarded as data preprocessing for model training. In other words, at the data curator side, *Solitude* can first proceed with the mechanism for feature vector denoising and then optimize Eq. 10 to obtain a "denoised" graph structure. After that, the "denoised" graph is treated as the input of the GNN model. By solving Eq. 13, the model parameters $\boldsymbol{\theta}$ can be obtained. Note that our ultimate goal is to obtain a GNN model with optimum generalization capability measured by the prediction accuracy on the test set. From this perspective, we can treat the process of graph structure denoising as a form of regularization, which is aligned with the regularization technique that prevents neural networks from overfitting [31]. Accordingly, the loss function for model training can be reformulated as:

$$\min_{\mathbf{A}^c, \boldsymbol{\theta}} \mathcal{L}_{\mathbf{GNN}}(\mathbf{A}^c, \mathbf{X}^c, \boldsymbol{\theta}) + \lambda_1 ||\tilde{\mathbf{A}} - \mathbf{A}^c||_{\mathbf{F}}^2 + \lambda_2 ||\mathbf{A}^c||_1, \tag{14}$$

where $\lambda_1$ and $\lambda_2$ control the regularization ratio. To solve Eq. 14, an alternating optimization scheme with Adam is used to iteratively update $\boldsymbol{\theta}$ and $\mathbf{A}^c$. As there is no parameter learning during feature denoising, the feature smoothing component will proceed before model training.

*5) Discussions:* In addition to introducing strong baselines for evaluating future improvements to private learning on graphs, our work suggests several open problems and directions for future work:

In this work, the notion of edge-LDP is built upon the randomized mechanism for the neighbor lists represented by a binary vector. This assumption implies that every user and the data curator know how many users and their indexes. We leave the problem of designing new mechanisms that satisfy edge-LDP when the users and the data curator may not know the entire set of the users as future work.

As illustrated in Section II-C, there are two variants of LDP when applying LDP to graph data: node-LDP and edge-LDP, both of which offer different kinds of privacy protection. In this work, we only consider edge-LDP and LDP for node feature vectors. We believe that private learning on graphs under the notion of node-LDP is a promising direction as well.

Differentially private transfer learning has been studied in prior work and has shown to be a natural candidate for privacy-preserving machine learning in various domains [10]. As illustrated in the work of Florian and Dan [11], the heuristic rule "better models transfer better" also holds with differential privacy. Therefore, differentially private graph learning with access to public data from a similar domain may be a feasible solution to improve the privacy-utility guarantees.

## IV. Evaluating Privacy-Preserving GNNs

### A. Datasets and Experimental Setup

*1) Datasets:* We conducted experiments over 3 datasets, falling into two categories: citation networks and social networks. For citation networks, we use Cora [29] and CiteSeer [30], both of which are benchmarking datasets for node classification. In these two datasets, nodes represent scientific publications, and edges correspond to the citation links. These two datasets contain bag-of-words feature vectors for each publication; each publication has a class label. For the real-world social networks, we use LastFM [32], which is a dataset collected from a music streaming service, in which nodes are users from Asian countries, and links represent friendships. Its task is to predict the home country of a user given the artists liked by them. As this dataset was highly imbalanced, for fair comparisons, we limit the classes to the top-10 ones with the most samples, as was done in [15]. The detailed statistics of the used datasets are described in Table IV.

*2) Baselines:* Our framework is the first effort towards private learning over decentralized network graphs to the best of our knowledge. We compare *Solitude* against the **Base** methods including GCN [12] and GraphSage [5]. GCN and GraphSage are two representative graph neural networks. By evaluating our approach with various GNN backbones, we can show that *Solitude* is able to preserve edge privacy and node feature privacy by incorporating with any GNNs. In particular, we use the same randomized mechanisms to obfuscate the feature vectors and adjacency lists. The main difference is that the base methods are directly trained over the noisy graphs without any calibration process. For better effectiveness demonstration, we also use an alternative baseline — **LPGNN** [15]. This method assumes that the data curator can access the global topology, and it was proposed to preserve the privacy of the node features. Though the setting of LPGNN and ours are different, to be comparable, we adapt LPGNN into our setting by randomizing the adjacency lists with our proposed mechanism and do not consider the labels of the training samples as the private information of the users. We argue that once the model parameters are achieved, the data curator can obtain the labels for any node by treating them as the model input as long as the model is well trained.

*3) Experimental Setup:* We follow the data preprocessing as reported in [15]. Specifically, for all datasets, we randomly split each dataset into three portions: 50% for training, 25% for validation, and 25% for the test. For the datasets with node features, including Cora, CiteSeer, and LastFM, the randomized mechanisms for LDP are applied to the node features and the adjacency lists of all training, validation, and test sets. All the GNN models, including the base methods and the backbone models of LPGNN and *Solitude*, consist of two graph convolution layers — each of which has a hidden dimension of size 16 — and a SeLU activation function [33] followed by dropout.

As for the evaluation metrics, we employ the standard metric — classification accuracy on the test set (or called prediction accuracy) under various privacy budgets — to evaluate the generalization capability of the learning model. Unless otherwise stated, all experiments are run 5 times to ensure statistical significance. Specifically, we report the mean and standard deviation values over 5 runs.

*Parameter settings.* For general hyperparameters, we applied a grid search to find the best choices: the learning rate, dropout rate, and weight decay were tuned among $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$, the feature smoothing steps $l_x$ and label smoothing steps $l_y$ were searched from $\{0,\ 2,\ 4,\ 8\}$, and the coefficients of $\lambda_1$ and $\lambda_2$ were searched in $\{10^{-5},\ 10^{-4},\ 10^{-3},\ 10^{-2}\}$. We used the Xavier initializer [34] and the Adam SGD optimizer [35] for all models. In addition, the maximum epoch was set as 500. For all $\epsilon_x$ and $\epsilon_a$ pairs, we traversed all the parameters to get its optimal performance in the experimental environment. Without specification, we report the results under the hyperparameters with the best performance overall. More specifically, we use the best learning rate, weight decay, and dropout for every $\epsilon_x$ and $\epsilon_a$ pairs. The selection rationality of the privacy budgets is discussed below.

### B. Experimental Results

We first investigate how *Solitude* performs under varying feature and edge privacy budgets. In particular, the privacy budget for the node features varies within $\{0.5, 1, 2\}$. The maximum value of $\epsilon_x$ is selected based on the proposition in [15], which indicates that in the high-privacy regime $\epsilon_x \leq 2.18$, the multi-bit mechanism perturbs at least one random dimension. Similarly, for randomized response flipping at least one edge of a node, the probability satisfies $p = \frac{1}{1+\mathrm{e}^{\epsilon_a}} \geq \frac{1}{|\mathbf{V}|}$, where $|\mathbf{V}|$ denotes the number of nodes in the graph. Then, we can obtain $\epsilon_a \leq \ln(N-1)$. Therefore, we varies the edge privacy budgets within $\{7.0, 7.3, 7.5, 7.7, 7.9, 8.0\}$ for two citation networks, and $\{7.7, 7.9, 8.0, 8.3, 8.5, 8.7, 8.9\}$ for LastFM. We first fix the privacy budget for the features and compare the performance under various edge privacy budgets. Table III reports the classification accuracy of different methods when $\epsilon_x = 1$.

We can observe that our proposed framework consistently achieves the best performance in most cases. Concretely, *Solitude* has an improvement by up to 9.3% on LastFM as compared to the base methods. As for the two citation networks, our framework achieves significant performance gain ranging from around 7.6% to 18.6% on Cora, and from 5.2% to 15.1% on Citeseer comparing to the base methods (see Table III). The difference in performance gains between LastFM and two citation networks implies that lower degree networks enjoy more benefits from our framework. The reason is that the randomized response has a higher impact on

TABLE III

CLASSIFICATION ACCURACY (%) WITH $\epsilon_x = 1$, ON VARIOUS VALUES OF $\epsilon_a$

| DATASET | | $\epsilon_a$ | 7.0 | 7.3 | 7.5 | 7.7 | 7.9 | 8.0 |
|---|---|---|---|---|---|---|---|---|
| CORA | GRAPHSAGE | BASE | 50.9 ± 3.8 | 54.9 ± 3.7 | 57.2 ± 2.5 | 61.4 ± 2.6 | 63.7 ± 2.4 | 63.2 ± 3.2 |
| | | LPGNN | 62.5 ± 1.6 | 69.2 ± 1.6 | 72.9 ± 2.3 | 74.7 ± 0.9 | 75.3 ± 1.2 | 76.4 ± 0.8 |
| | | *Solitude* | **66.4** ± 1.7 | **72.4** ± 1.2 | **75.8** ± 0.9 | **76.7** ± 1.1 | **77.9** ± 0.6 | **79.0** ± 0.6 |
| | GCN | BASE | 60.8 ± 2.0 | 63.1 ± 3.4 | 64.8 ± 3.7 | 66.9 ± 4.0 | 68.1 ± 4.0 | 68.6 ± 3.8 |
| | | LPGNN | 67.9 ± 1.2 | 69.4 ± 1.4 | 73.7 ± 1.8 | 74.4 ± 0.5 | 75.7 ± 1.0 | 76.3 ± 0.9 |
| | | *Solitude* | **68.4** ± 2.3 | **72.6** ± 1.4 | **75.2** ± 1.2 | **76.1** ± 0.5 | **77.3** ± 0.9 | **77.8** ± 0.5 |
| CITESEER | GRAPHSAGE | BASE | 35.0 ± 1.8 | 39.1 ± 1.3 | 42.3 ± 1.5 | 44.5 ± 1.2 | 45.7 ± 1.8 | 44.5 ± 1.9 |
| | | LPGNN | 46.8 ± 2.0 | 49.7 ± 0.7 | 50.7 ± 1.4 | 51.2 ± 1.4 | 53.3 ± 0.9 | 54.5 ± 1.2 |
| | | *Solitude* | **50.1** ± 1.3 | **52.7** ± 1.0 | **54.0** ± 1.0 | **55.6** ± 0.8 | **56.3** ± 1.0 | **57.0** ± 1.4 |
| | GCN | BASE | 39.7 ± 3.0 | 45.3 ± 1.8 | 48.3 ± 2.9 | 49.6 ± 1.3 | 52.5 ± 2.1 | 52.8 ± 2.5 |
| | | LPGNN | 47.8 ± 1.1 | 50.1 ± 1.8 | 52.3 ± 1.2 | 53.7 ± 1.3 | 55.6 ± 1.3 | 55.6 ± 1.5 |
| | | *Solitude* | **49.8** ± 1.4 | **53.4** ± 1.0 | **55.1** ± 0.8 | **56.4** ± 1.5 | **58.2** ± 0.7 | **58.0** ± 2.5 |
| LASTFM | | $\epsilon_a$ | 7.7 | 7.9 | 8.0 | 8.3 | 8.5 | 8.7 |
| | GRAPHSAGE | BASE | 62.6 ± 2.4 | 64.4 ± 1.8 | 66.5 ± 1.9 | 71.2 ± 1.9 | 72.0 ± 2.0 | 75.5 ± 1.4 |
| | | LPGNN | 62.6 ± 2.4 | 65.3 ± 5.8 | 68.3 ± 2.4 | 73.3 ± 3.2 | 74.6 ± 3.0 | 76.5 ± 3.1 |
| | | *Solitude* | **65.5** ± 7.1 | **69.4** ± 5.4 | **71.6** ± 2.4 | **77.2** ± 1.7 | **77.2** ± 1.4 | **79.0** ± 1.5 |
| | GCN | BASE | 60.5 ± 2.5 | 63.2 ± 2.1 | 63.8 ± 2.0 | 66.9 ± 1.1 | 68.5 ± 1.5 | 69.6 ± 1.6 |
| | | LPGNN | **62.3** ± 7.2 | 68.2 ± 0.8 | **68.3** ± 3.0 | 73.3 ± 2.4 | 76.9 ± 2.9 | **79.5** ± 0.8 |
| | | *Solitude* | 60.9 ± 2.6 | **68.9** ± 2.1 | 68.0 ± 4.7 | **75.1** ± 1.7 | **77.5** ± 1.9 | 78.9 ± 1.3 |

TABLE IV

STATISTICAL DESCRIPTION OF USED DATASETS

| DATASET | #NODES | #EDGES | #CLASSES | #FEATURES | AVG. DEGREE |
|---|---|---|---|---|---|
| CORA | 2,708 | 5,278 | 7 | 1,433 | 3.90 |
| CITESEER | 2,110 | 3,668 | 6 | 3,703 | 2.74 |
| LASTFM | 7,083 | 25,814 | 10 | 7,842 | 7.29 |

the graphs with more sparseness. Though LPGNN achieves comparable performance gains by comparing with the base methods, it still falls behind our framework in most cases. Note that LPGNN was designed specifically to protect the privacy of node features, while our framework can protect the node features and graph structure information simultaneously. The performance gain achieved by *Solitude* upon LPGNN indicates that the denoising component for graph structure is indeed effective in reducing the effects of introduced noise due to the randomized flipping.

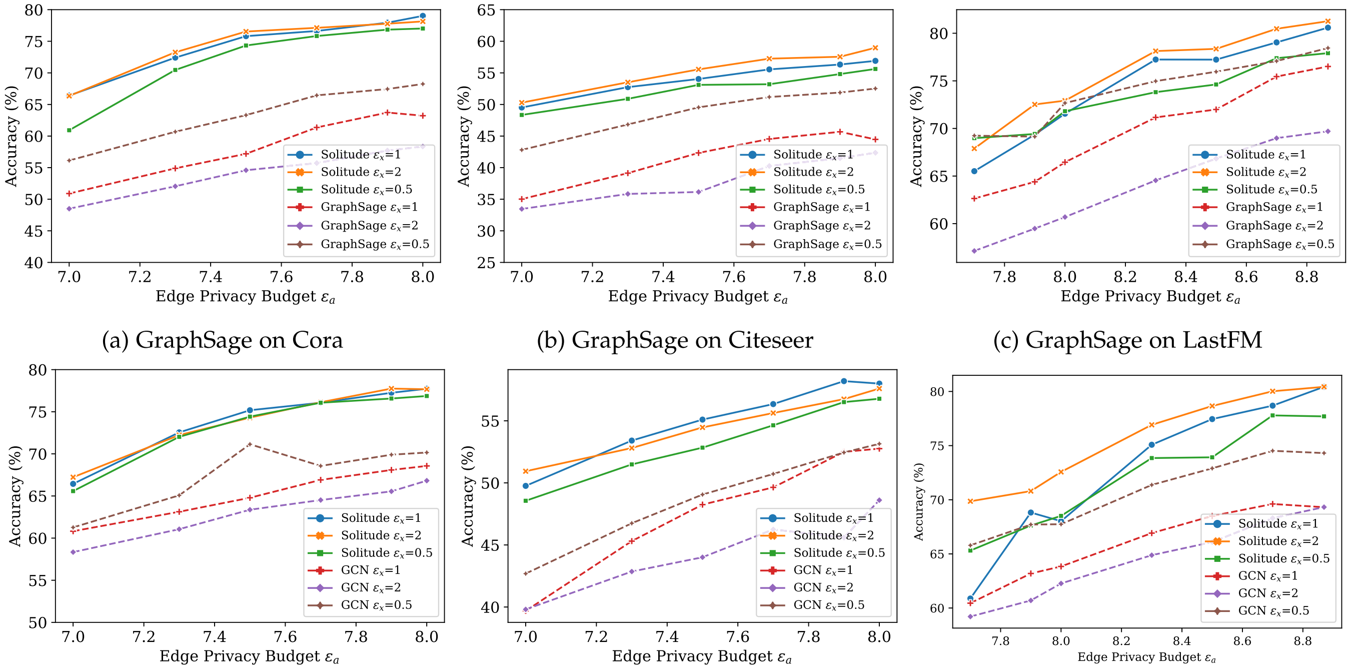
To further examine the effects of introduced noise in the feature vectors, we evaluate the performance of our framework with various privacy budgets for the node features and the adjacency lists. Fig. 4 shows the classification accuracy with different GNN architectures across three datasets. We see *Solitude* achieves better performance in the lower-privacy regime (with higher privacy budgets). This can be explained by the fact that the predictions become more accurate due to an increase in the privacy budget. Interestingly, we also observe that *Solitude* achieves better performance gains with higher privacy budgets for node features when the edge privacy budget is fixed. The maximum performance gain is different across datasets and privacy budgets. This result indicates that our denoising mechanisms effectively mitigate the overfitting issue caused by the introduced noise for privacy protection, improving the model utility.

Essentially, for GNNs to work, the *smoothness assumption* has to hold. As the crux of GNNs for node classification is to propagate features and labels throughout the network graph. Inspired by this, we are interested in investigating how the label rates affect the model utility with local differential privacy. In particular, label rate denotes the number of labeled nodes for training divided by the total number of nodes in each dataset. Fig. 5 depicts the classification accuracy under various label rates. It shows that the prediction accuracy consistently increases with the increase of the label rates over all datasets. We can conclude that improving privacy-utility guarantees needs more labeled nodes for training.

## V. RELATED WORK

### A. Privacy-Preserving Analysis of Graph Statistics

Differential privacy (DP) has emerged as a de-facto standard for privacy guarantees [7], [25]. Most existing work focuses on centralized differential privacy [36], [37], and they are specialized for analyzing the graph statistics, such as degree distribution estimation [27], subgraph counts [36]. LDP is a special case of differential privacy in the local model. Each participant obfuscates their data portion locally and then sends the obfuscated data to a data curator (possibly a malicious



(a) GraphSage on Cora

(b) GraphSage on Citeseer

(c) GraphSage on LastFM

(d) GCN on Cora

(e) GCN on Citeseer

(f) GCN on LastFM

Fig. 4. Classification accuracy with different GNN architectures. The horizontal axes are different values of edge privacy budgets. The vertical axes are the prediction accuracy of the test set. *Solitude* obtains significance gains on Cora, Citeseer, and LastFM on various values of $\epsilon_x$ and $\epsilon_a$.

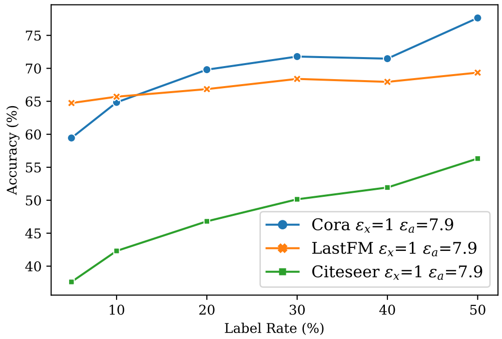


Fig. 5. Classification accuracy under various label rates. It shows that improving privacy-utility guarantees needs more labeled samples.

third-party). Prior works on graph data with LDP mainly focus on the estimations of graph statistics, such as clustering coefficient estimation [9], heavy hitter estimation [13], and frequency estimation [14].

Among others, some works are specialized for subgraph counts [8], [38], such as triangles, 3-hop paths, and $k$-cliques. Specifically, Sun *et al.* [38] proposes a multi-phase framework under decentralized differential privacy, which assumes that each user/data holder is aware of not only her connections but also a broader subgraph in her local neighborhood. Subsequently, Imola *et al.* [8] propose new algorithms for triangle and $k$-star counts, assuming that each user can only access her connections. LDPGen [17] was proposed to generate synthetic graphs in the setting where a data curator collects subgraphs from the participants under the notion of edge differential privacy.

## B. Privacy Leakage on GNNs

There exists some prior work on the privacy implication of training GNNs on graphs [39], [40], [41]. Duddu em *et al.* [40] analyzed the privacy risks of graph embedding with various attacks, including membership inference attacks, graph reconstruction attacks, and attribute inference attacks. Later, Olatunji em *et al.* [39] proposed membership inference attacks under restricted scenarios, by 0-hop query and 2-hop query to the victim models. He em *et al.* [41] showed that given various background knowledge, the outputs of a GNN indeed could be utilized to infer the rich information about the graph structure used to train the model. They coined the attack “link stealing attack.”

## C. Privacy-Preserving Graph Learning With GNNs

There is a research line that attempts to address privacy in graph learning with federated learning [42], [43] and split learning, which is orthogonal to our setting. The closest to ours is LPGNN [15], wherein a set of mechanisms are proposed to protect the privacy of node features. Specifically, LPGNN assumes that a central server holds the global graph topology, and the server is allowed to collect the node features satisfying local differential privacy. It strives to preserve the privacy of node features while maintaining the generalization capability of the learned GNNs. However, the assumption of holding the global topology in the central server hinders its practicality in many real-world applications, where the privacy of the graph topology is of paramount importance. Here we fill this gap and show the effectiveness of our privacy-preserving learning framework based on graph neural networks, with local differential privacy guarantees.

## VI. Concluding Remarks

In this paper, we propose a new privacy-preserving learning framework for decentralized network graphs based on graph neural networks, called *Solitude*. It can simultaneously preserve edge privacy and node feature privacy for every user and seamlessly incorporate with any GNN architectures, such as GCN and GraphSage, with privacy-utility guarantees. The key of *Solitude* is a set of new mechanisms that can calibrate the introduced noise in the decentralized graph to mitigate the overfitting problem while learning over noisy graphs. Theoretical analysis and extensive experiments on benchmarks have demonstrated the rationality and effectiveness of our proposed mechanisms as compared to the alternative baselines.

Due to the inherent limitations of current differential privacy formalisms for graph learning, our design needs a certain amount of privacy budget to have a comparative learning performance. We hope our study pushes forward the progress on privacy-preserving GNNs under local differential privacy. There are many possible directions for future work, such as establishing novel formalisms of local differential privacy, i.e., edge differential privacy, specifically for graph learning. We leave this in our future work.

## Acknowledgment

The assistance provided by Yuteng Sun from The Chinese University of Hong Kong was greatly appreciated.

## References

[1] K. Rusek, J. Suarez-Varela, P. Almasan, P. Barlet-Ros, and A. Cabellos-Aparicio, “RouteNet: Leveraging graph neural networks for network modeling and optimization in SDN,” *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2260–2270, Oct. 2020.

[2] M. Zitnik, M. Agrawal, and J. Leskovec, “Modeling polypharmacy side effects with graph convolutional networks,” *Bioinformatics*, vol. 34, no. 13, pp. i457–i466, Jul. 2018.

[3] M. Zitnik and J. Leskovec, “Predicting multicellular function through multi-layer tissue networks,” *Bioinformatics*, vol. 33, no. 14, p. i190, Jul. 2017.

[4] J. Zhou, Z. Xu, A. M. Rush, and M. Yu, “Automating botnet detection with graph neural networks,” in *Proc. AutoML Netw. Syst. Workshop MLSys Conf.*, 2020.

[5] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017.

[6] F. Wu, Y. Long, C. Zhang, and B. Li, “LINKTELLER: Recovering private edges from graph neural networks via influence analysis,” in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2022.

[7] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, “Local privacy and statistical minimax rates,” in *Proc. IEEE 54th Annu. Symp. Found. Comput. Sci.*, Oct. 2013.

[8] J. Imola, T. Murakami, and K. Chaudhuri, “Locally differentially private analysis of graph statistics,” in *Proc. Secur. Symp. (USENIX Secur.)*, 2021.

[9] Q. Ye, H. Hu, M. H. Au, X. Meng, and X. Xiao, “LF-GDPR: A framework for estimating graph metrics with local differential privacy,” *IEEE Trans. Knowl. Data Eng.*, early access, Dec. 24, 2021, doi: 10.1109/TKDE.2020.3047124.

[10] M. Abadi *et al.*, “Deep learning with differential privacy,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2016.

[11] T. Florian and B. Dan, “Differentially private learning needs better features (or much more data),” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021.

[12] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017.

[13] Z. Qin, Y. Yang, T. Yu, I. Khalil, X. Xiao, and K. Ren, “Heavy hitter estimation over set-valued data with local differential privacy,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016.

[14] T. Wang, J. Blocki, N. Li, and S. Jha, “Locally differentially private protocols for frequency estimation,” in *Proc. Secur. Symp. (USENIX Secur.)*, 2017.

[15] S. Sajadmanesh and D. Gatica-Perez, “Locally private graph neural networks,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2021.

[16] D. Konforty, Y. Adam, D. Estrada, and L. G. Meredith, “Synereo: The decentralized and distributed social network,” Tech. Rep., 2015.

[17] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, “Generating synthetic decentralized social graphs with local differential privacy,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017.

[18] C. Dwork and A. Roth, “The algorithmic foundations of differential privacy,” *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, Aug. 2014.

[19] B. Ding, J. Kulkarni, and S. Yekhanin, “Collecting telemetry data privately,” in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017.

[20] D. Zhu, Z. Zhang, P. Cui, and W. Zhu, “Robust graph convolutional networks against adversarial attacks,” in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019.

[21] Y. Bengio, O. Delalleau, and N. Le Roux, “Label propagation and quadratic criterion,” Tech. Rep., 2006.

[22] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, “Learning with local and global consistency,” in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2004.

[23] Tencent. *Wechat: Connecting a Billion People with Calls, Chats, and More*. [Online]. Available: https://www.wechat.com

[24] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A comprehensive survey on graph neural networks,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Apr. 2020.

[25] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, “What can we learn privately?” in *Proc. 49th Annu. IEEE Symp. Found. Comput. Sci.*, Oct. 2008.

[26] J. Blocki, A. Blum, A. Datta, and O. Sheffet, “The Johnson-Lindenstrauss transform itself preserves differential privacy,” in *Proc. IEEE 53rd Annu. Symp. Found. Comput. Sci.*, Oct. 2012.

[27] W.-Y. Day, N. Li, and M. Lyu, “Publishing graph degree distribution with node differential privacy,” in *Proc. Int. Conf. Manage. Data*, Jun. 2016.

[28] J. You, J. M. Gomes-Selman, R. Ying, and J. Leskovec, “Identity-aware graph neural networks,” in *Proc. AAAI Conf. Artif. Intell.*, 2021.

[29] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, “Automating the construction of internet portals with machine learning,” *Inf. Retr.*, vol. 3, no. 2, pp. 127–163, 2000.

[30] C. L. Giles, K. D. Bollacker, and S. Lawrence, “CiteSeer: An automatic citation indexing system,” in *Proc. 3rd ACM Conf. Digit. Libraries*, 1998.

[31] J. Kukavcka, V. Golkov, and D. Cremers, “Regularization for deep learning: A taxonomy,” 2017, *arXiv:1710.10686*.

[32] B. Rozemberczki and R. Sarkar, “Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models,” in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2020.

[33] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, “Self-normalizing neural networks,” in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017.

[34] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proc. Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

[35] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015.

[36] V. Karwa, S. Raskhodnikova, A. Smith, and G. Yaroslavtsev, “Private analysis of graph structure,” *Proc. VLDB Endowment*, vol. 4, no. 11, pp. 1146–1157, 2011.

[37] Y. Wang, X. Wu, and L. Wu, “Differential privacy preserving spectral graph analysis,” in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining*. Cham, Switzerland: Springer, 2013.

[38] H. Sun *et al.*, “Analyzing subgraph statistics from extended local views with decentralized differential privacy,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019.

[39] 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.

[40] V. Duddu, A. Boutet, and V. Shejwalkar, "Quantifying privacy leakage in graph embedding," in *Proc. 17th EAI Int. Conf. Mobile Ubiquitous Syst., Comput., Netw. Services (MobiQuitous)*, Dec. 2020, pp. 1–11.

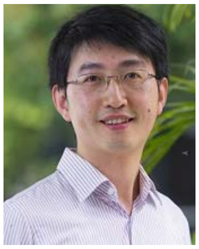
[41] X. He, J. Jia, M. Backes, N. Z. Gong, and Y. Zhang, "Stealing links from graph neural networks," in *Proc. 30th Secur. Symp. (USENIX Secur.)*, 2021.

[42] C. Meng, S. Rambhatla, and Y. Liu, "Cross-node federated graph neural network for spatio-temporal data modeling," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021.

[43] C. He *et al.*, "FedGraphNN: A federated learning system and benchmark for graph neural networks," in *Proc. Workshop Graph Neural Netw. Syst.*, 2021.

**Wanyu Lin** (Member, IEEE) received the B.Engr. degree from the School of Electronic Information and Communications, Huazhong University of Science and Technology, China, the M.Phil. degree from the Department of Computing, The Hong Kong Polytechnic University, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Toronto. Her research interests include graph machine learning, trustworthy machine learning, data privacy, and model interpretability. She has served as an Associate Editor for IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (TNNLS).

**Baochun Li** (Fellow, IEEE) received the B.Engr. degree from the Department of Computer Science and Technology, Tsinghua University, China, in 1995, and the M.S. and Ph.D. degrees from the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, in 1997 and 2000. Since 2000, he has been with the Department of Electrical and Computer Engineering, University of Toronto, where he is currently a Professor. He has also been holding the Bell Canada Endowed Chair in computer engineering since August 2005. His research interests include cloud computing, distributed systems, datacenter networking, and wireless systems. He is a member of ACM. He was a recipient of the IEEE Communications Society Leonard G. Abraham Award in the Field of Communications Systems in 2000. In 2009, he was a recipient of the Multimedia Communications Best Paper Award from the IEEE Communications Society and the University of Toronto McLean Award.



**Cong Wang** (Fellow, IEEE) is currently a Professor with the Department of Computer Science, City University of Hong Kong. His research interests include data and network security, blockchain and decentralized applications, and privacy-enhancing technologies. He is a member of ACM. He has been one of the Founding Members of the Young Academy of Sciences of Hong Kong since 2017, and has been conferred the RGC Research Fellow in 2021. He received the Outstanding Researcher Award in 2019, the Outstanding Supervisor Award in 2017, and the President's Awards in 2019 and 2016 from City University of Hong Kong. He was a co-recipient of the Best Paper Award of IEEE ICDCS 2020, ICPADS 2018, MSN 2015, the Best Student Paper Award of IEEE ICDCS 2017, and the IEEE INFOCOM Test of Time Paper Award 2020. His research has been supported by multiple government research fund agencies, including the National Natural Science Foundation of China, Hong Kong Research Grants Council, and Hong Kong Innovation and Technology Commission. He has served as an Associate Editor for IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING (TDSC), IEEE TRANSACTIONS ON SERVICES COMPUTING (TSC), IEEE INTERNET OF THINGS JOURNAL (IoT-J), IEEE NETWORKING LETTERS, and *Journal of Blockchain Research*, and the TPC Co-Chair for a number of IEEE conferences and workshops.