Heterogeneous Randomized Response for Differential Privacy in Graph Neural Networks

Khang Tran, Phung Lai, NhatHai Phan*
New Jersey Institute of Technology, USA
{kt36, tl353, phan}@njit.edu

Issa Khalil
Qatar Computing Research Institute, Qatar
ikhalil@hbku.edu.qa

Yao Ma, Abdallah Khreishah
New Jersey Institute of Technology, USA
{yao.ma, abdallah}@njit.edu

My T. Thai
University of Florida, USA
mythai@cise.ufl.edu

Xintao Wu
University of Arkansas, USA
xintaowu@uark.edu

***Abstract*—Graph neural networks (GNNs) are susceptible to privacy inference attacks (PIAs) given their ability to learn joint representation from features and edges among nodes in graph data. To prevent privacy leakages in GNNs, we propose a novel heterogeneous randomized response (HETERORR) mechanism to protect nodes' features and edges against PIAs under differential privacy (DP) guarantees, without an undue cost of data and model utility in training GNNs. Our idea is to balance the importance and sensitivity of nodes' features and edges in redistributing the privacy budgets since some features and edges are more sensitive or important to the model utility than others. As a result, we derive significantly better randomization probabilities and tighter error bounds at both levels of nodes' features and edges departing from existing approaches, thus enabling us to maintain high data utility for training GNNs. An extensive theoretical and empirical analysis using benchmark datasets shows that HETERORR significantly outperforms various baselines in terms of model utility under rigorous privacy protection for both nodes' features and edges. That enables us to defend PIAs in DP-preserving GNNs effectively.**

***Index Terms*—differential privacy, GNNs, privacy inference**

I. INTRODUCTION

Graph Neural Networks (GNNs) have been well-known for their ability to learn from graph data, simultaneously leveraging the nodes' features and the graph structure [1]. However, GNNs are vulnerable to PIAs since the nodes' features and the edges often contain sensitive information of the participants, which can be inferred by the adversaries when GNNs are deployed [2]. Attacks such as membership inference [3], and structure inference [4] underline privacy risks in GNNs. Hence, to promote the broader adoption of GNNs, it is essential to protect graph data privacy in training GNNs while maintaining high model utility.

Among privacy preserving techniques, differential privacy (DP), a rigorous formulation of privacy in probabilistic terms without computational overhead, is one of the golden standards. DP has been applied to protect either edge privacy [5] or nodes' feature privacy [6] given graph data. To protect both the graph structure and node features, a straightforward approach to achieve DP protection at both nodes' feature-level and graph structure-level in training GNNs is applying both node-feature and graph-structure DP-preserving mechanisms independently. However, that treatment can significantly degrade the graph data utility resulting in poor model performance, especially in the application of GNNs. This is a challenging and open problem since a minor privacy-preserving perturbation to either features of a single node or a local graph structure will negatively affect its neighbors. In addition, the impact is propagated through the entire graph. There are two main reasons for this problem: first, the correlation between the nodes is very high, therefore, quantifying the privacy risk through the aggregation of GNNs is intractable and adding a little noise to one node can impact all of its neighbors; second, most of graphs in practice are sparse and adding noise to the structure of the graph can easily destroy the sparsity of it, resulting in low graph structure utility.

Key Contributions. To address this problem, we develop a new heterogeneous randomized response (HETERORR) mechanism to preserve nodes' features and edges privacy for graph data in GNNs application. Our methods are based on randomize response (RR) [7], [8] which is an advanced and effective method for privacy-preserving. HETERORR leverages the heterogeneity in graph data to optimize the magnitude of privacy preserving-noise injected into nodes' features and the graph structure, such that ***less sensitive*** and ***more important*** features (to the model outcome) and edges receive ***lower probabilities to be randomized (less noisy)***, and vice versa. This property of HETERORR enables us to achieve significantly better utility compared with homogeneous randomization probabilities in existing mechanisms under the same privacy guarantee.

Furthermore, HETERORR is applied in the pre-processing step to create a privacy-preserving graph that can be stored and reused as a replacement for the original graph. Due to the post-processing of DP, every analysis on the privacy-preserving graph satisfies the DP guarantee for the original graph, which makes HETERORR a permanent RR. Therefore, it provides longitudinal DP protections without accumulation of privacy risks over the time.

An extensive theoretical and empirical analysis conducted

* *Corresponding author*

978-1-6654-8045-1/22/$31.00 ©2022 IEEE

on benchmark datasets employing GNNs as a motivating application shows that HETERORR significantly outperforms baseline approaches in terms of data and model utility under the same privacy protection. Importantly, HETERORR are resilient against PIAs by reducing the attack success rate to a random guess level without affecting the GNNs' model utility. Our implementation and supplemental documents can be found here: *https://github.com/khangtran2020/DPGNN.git*

II. BACKGROUND

This section provides an overview of GNNs, privacy threat models, and existing defenses.

a) Graph Learning Setting: A service provider possesses a private graph $G(\mathcal{V}, \mathcal{E})$ constructed from its users' data, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges. Each node $v \in \mathcal{V}$ has its raw (data) input x and a *ground-truth* one hot vector $y \in \{0,1\}^{\mathcal{C}}$ with $\mathcal{C}$ is the number of output classes. Each node (user) $v \in \mathcal{V}$ has a set of *public* (non-sensitive) edges and a set of *private* (sensitive) edges (i.e., $\mathcal{E} = \mathcal{E}_{pub} \cup \mathcal{E}_{pri}$). This is a practical setting in many real-world applications.

For instance, in a FLICKR network [9], an edge between a pair of images (nodes) can be created by a mutual friend connection of the image owners. These edges can expose private friend connections among the image owners; thus, they are sensitive and need to be protected. Meanwhile, an edge between a pair of images constructed based on either shared galleries or common tags is considered public since the edge does not expose private connections among the image owners.

In practice, one can use a pretrained model $g(\cdot)$ to extract a d-dimension embedding vector $z = g(x)$ as an initial representation of each node $v \in \mathcal{V}$. A K-layer GNN learns the embedding representation for each node $v \in \mathcal{V}$ through a stack of K graph convolutional layers. Each layer $k \in [1, K]$ takes as input the embedding $h_v^{(k-1)}$ for $v \in \mathcal{V}$ from the previous layer, then updates the embedding as follows:

$$h_{\mathcal{N}(v)}^{(k)} = \text{AGG}\left(h_v^{(k-1)} \cup \{h_u^{(k-1)}, u \in \mathcal{N}(v)\}\right) \quad (1)$$

$$h_v^{(k)} = \sigma\left(W^{(k)} h_{\mathcal{N}(v)}^{(k)}\right) \quad (2)$$

where $\mathcal{N}(v)$ is the neighborhood of node v, $h_v^{(0)} = z_v$, $\text{AGG}(\cdot)$ is an aggregation function, $W^{(k)}$ is the trainable parameters of layer k, and $\sigma(\cdot)$ is a non-linear activation function.

In this work, we consider a node classification task, in which each node is classified into one of the output classes. At the inference time, the service provider releases APIs to query the trained model in applications. This is a practical setting of ML-as-a-service (MLaaS) for GNNs [4].

b) Privacy Threat Models: Given the graph learning setting, we consider the threat model as in Figure 1. An adversary aims to infer the nodes' raw (data) input and the private connections in the private graph G. Firstly, the adversary collects the auxiliary information of the nodes' features and the public edges from the public sources to infer the private connections by conducting the LinkTeller attack [4]. Secondly, the adversary uses the auxiliary information with the inferred

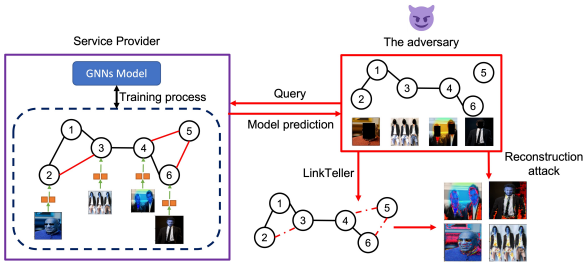


Fig. 1. Privacy threat

set of edges to perform an inference attack to infer nodes' embedding features [10]. Finally, the adversary uses the inferred embedding features to reconstruct the raw (data) input. This threat model leads to severe privacy leakages in using sensitive graph data in GNNs.

c) Differential Privacy (DP) [11]: DP is a privacy-aware computational approach that assure the output of a mechanism is not strongly dependent on a particular data-point. However, in graph data, features of a particular node can be inferred by neighboring nodes' features. To address this problem, one can employ local DP (LDP) to ensure that the privacy-preserving randomization of every each node's features is independently its neighboring nodes. The definition of LDP is as follows:

Definition 1 (Local DP)**.** A randomized algorithm $\mathcal{M}$ satisfies ε-LDP if and only if for any pair of inputs z, z' in the input space, and the output space $S \subseteq Range(\mathcal{M})$, it satisfies

$$Pr[\mathcal{M}(z) \in S] \leq e^{\varepsilon} Pr[\mathcal{M}(z') \in S] \quad (3)$$

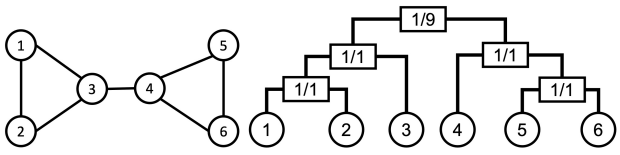
where ε is the privacy budget. The privacy budget ε control how the output distribution conditioned by z and z' may differ. A smaller value of ε ensure a better privacy protection.

One of effective methods to preserve LDP is applying randomized response (RR) mechanisms [7], [8]. Existing RR methods consider a homogeneous scenario where every features in the input space have the same sensitivity and importance which is not utility-optimal since the input space is heterogeneous in most of real-world tasks. Departing from existing approaches, we derive heterogeneous randomization probabilities across features by balancing their sensitivity and importance; thus achieving better graph data privacy-data utility trade-offs in HETERORR.

d) DP Preservation in Graph Structure: DP preservation in graph analysis can be generally categorized into node-level DP [12] and edge-level DP [13]. Node-level DP and edge-level DP aims to protect the presence of the set of nodes or edges, respectively. In this work, we focus on edge-level DP to protect the privacy of graph structure, defined as follows:

Definition 2 (Edge-level DP [13])**.** A randomized algorithm $\mathcal{M}$ satisfies edge-level ε-DP if for any two neighboring graphs $G(\mathcal{V}, \mathcal{E})$ and $G'(\mathcal{V}, \mathcal{E}')$ differ in one edge while they share the same set of nodes and an output space $\mathcal{O} \subseteq Range(\mathcal{M})$,

$$P(\mathcal{M}(G) \in \mathcal{O}) \leq e^{\varepsilon} P(\mathcal{M}(G') \in \mathcal{O}) \quad (4)$$



(a) Graph $G(\mathcal{V}, \mathcal{E})$ (b) HRG model $(\mathcal{D}, \{p_r\})$

Fig. 2. An instance of HRG model (b) represents a given graph G (a). Considering the common ancestor of the leaves (1) and (2): since there are one leaf in the left child and one leaf in the right child, the number of possible edges is one. Since there are one edges between node (1)-(2) in G, the probability p_r at this internal node r is 1/1.

where ε is the privacy budget. Edge-level DP ensures that the adversary cannot infer the existence of a targeted edge with high confidence.

Previous edge-level mechanisms consider the whole set of edges is private. That may not be practical in certain real-world scenarios. Therefore, HETERORR focuses on protecting the set of private edges and leverages the information from the public edges to optimize the model utility.

e) Hierarchical Random Graph (HRG) [14]: To design an edge-level DP-preserving mechanism, one of the state-of-the-art approaches is representing the given graph G as a HRG model [14]. HRG is a statistical inference model that represents a hierarchical property of a given graph G by a binary tree dendrogram D as illustrated in Figure 2. In the dendrogram D, the number of leaves equal to the number of nodes in G. Each internal node $r \in D$ is associated with a probability of having a connection between the left and right child of r. Clauset et al. [14] proposed using Monte Carlo Markov Chain (MCMC) sampling to find the best HRG model to present a given graph. Xiao et al. [15] proposed to use the exponential and Laplace mechanisms to randomize the sampling process of HRG under DP. Then the DP-preserving HRG is used to sample a DP-preserving graph $\bar{G}$ which is released to the public. Different from [15], HETERORR leverages public edges to optimize the structural utility while providing edge-level DP guarantees to protect private edges. We achieve these two objectives in a unified edge-level DP-preserving MCMC sampling algorithm.

III. HETERORR: DP PRESERVING IN GNNS

In this section, we formally introduce HETERORR mechanism to preserve private information of both the nodes' embedding features and the private edges against the aforementioned threat model while maintaining high data utility.

Overview of HETERORR. HETERORR consists of two main components feature-aware randomized response (FEATURERR) and edge-aware randomized response (EDGERR) which provide nodes' feature-level and edge-level privacy protection respectively. First, FEATURERR randomizes every embedding feature to generate ε_f-LDP-preserving embedding features for every node. Second, EDGERR represents a given graph with an HRG model under DP protection and then uses the HRG model to sample an ε_e-DP-preserving graph as a replacement for the original graph. Finally, we combine the ε_f-LDP-preserving embedding features and the ε_e-DP-preserving graph structure to create a final and permanent DP preserving graph for training GNNs.

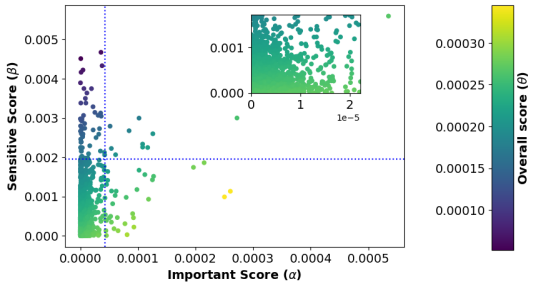


Fig. 3. The overall score θ with $\gamma = 0.5$

To optimize the data utility, first, FEATURERR balances the sensitivity and importance of each feature and randomize every feature such that *less sensitive* and *more important* features have *lower probabilities of being randomized* and vice versa. Secondly, in EDGERR, we propose a novel HRG model to capture the hierarchy of public and private edges in the original graph called and construct the PHRG by iteratively applying a new MCMC sampling process in which (1) we add noise to the MCMC to protect the private edges; and (2) we leverage public edges to preserve the original graph structure. As a result, HETERORR achieves better data utility under DP protection.

A. Feature-aware Randomized Response (FEATURERR)

a) Sensitivity and Importance: Let us present our method to determine the sensitivity and importance of the embedding features. In the input x, some input features are more sensitive or more important to the model outcome than others. This triggers a simple question: *"How could we quantify the sensitivity and importance of an embedding feature given an input feature?"* To answer this question, first, we quantify the sensitivity of an embedding feature as the maximal magnitude it can be changed by completely removing the sensitive input features from the input x. Therefore, we mask all of the sensitive input features out in a masked version $\hat{x}$ and extract its embedding features $\hat{z} = g(\hat{x})$. Then, we quantify the sensitivity β_i of an embedding feature i as follows:

$$\forall i \in [d] : \beta_i = \frac{|z_i - \hat{z}_i|}{\|z - \hat{z}\|_1} \tag{5}$$

Regarding the importance of the embedding feature i, denoted α_i, we employ the SHAP metric [16], one of the most well-applied model explainers, to quantify the influence of feature i on a model's decision. The importance score is quantified as follows:

$$\forall i \in [d] : \alpha_i = \frac{|SHAP(z_i)|}{\|SHAP(z)\|_1} \tag{6}$$

where $SHAP(z_i)$ is the SHAP score of the embedding feature z_i and $SHAP(z)$ is a vector of the SHAP scores for all the embedding features in z. In practice, we can compute SHAP scores for the embedding features by using a pretrained model that is trained on a publicly available dataset to avoid any extra privacy risks. Figure 3 shows the distribution of sensitivity and importance scores across embedding features.

To capture the correlation between sensitivity and importance scores, we define a unifying score θ_i by a linear combination of α_i and β_i, as follows:

$$\theta_i = \gamma\alpha_i + (1-\gamma)\big[\beta_{min} + (\beta_{max} - \beta_i)\big] \tag{7}$$

where $\gamma \in [0,1]$ is a weighted parameter to balance between the sensitivity score β_i and the importance score α_i, $\beta_{min} = \min_{j\in[d]} \beta_j$, $\beta_{max} = \max_{j\in[d]} \beta_j$, The idea of Eq. 7 is to separate between two set of features: the features have small values of α_i and high values of β_i (top-left corner of Figure 3); and the features have high values of α_i and small values of β_i (bottom-right corner in Figure 3) For the other features, since they are both more (less) important and more (less) sensitive (located within the top-right and bottom-left corner in Figure 3), Eq. 7 will smoothly combine the important and sensitive scores as a trade-off between importance and sensitiveness through the hyper-parameter γ

b) Randomizing Process: Given the unifying score θ_i, we randomize the feature i such that more important and less sensitive features (higher values of θ_i) will have higher probabilities to stay the same, and vice versa. That enables us to achieve better data utility. To achieve our goal, we assign a heterogeneous privacy budget $\varepsilon_i = \varepsilon_f \theta_i$ to the feature i. Then, we randomize each feature such that we provide ε_i-LDP for feature i while addressing the privacy-utility trade-off. We tackle this by minimizing the difference between the original and randomized value of feature i as follows.

Without loss of generality, considering the value of each feature is in the domain $[0,1]$. To optimize the data utility, we design a randomizing process such that the randomized values should fall in the original domain and the values nearer to the original value will have higher sampling probability. However, by considering the continuous domain, the sampling probability of each point in the domain is minuscule, therefore, we discretize the domain $[0,1]$ by k bins, resulting in a discrete domain $\{\frac{1}{k}, \ldots, 1\}$. This discretizing process will limit the outcomes, leading to a higher probability for each value. Then, we transform the value of each embedding feature from the $[0,1]$ domain to the discrete domain. Formally, the value z_i of embedding feature i will have the value $\frac{t}{k}$ if $\frac{t-1}{k} \leq z_i \leq \frac{t}{k}, t \in \{1,2,\ldots,k\}$. This can done as a data preprocessing step without extra privacy risks.

We randomize the value of each feature by the following rule: Given the value of the embedding feature i is $z_i = \frac{t}{k}$, we randomize embedding feature i such that it will have a randomized value $\frac{u}{k}, \forall u \in \{1,\ldots,k\}$, with the probability

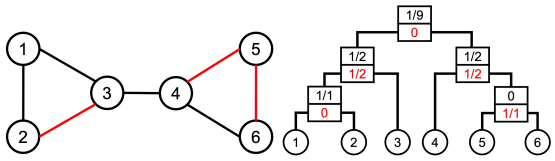
$$Pr\Big(\frac{u}{k}\Big|\frac{t}{k}\Big) = \frac{1}{C_t}\exp\Big(-\frac{|u-t|}{k\sigma_i}\Big) \tag{8}$$

where σ_i is the noise scale of feature i and C_t is the normalization parameter and quantified by:

$$C_t = \sum_{u=1}^{k} \exp\Big(-\frac{|u-t|}{k\sigma_i}\Big) \tag{9}$$

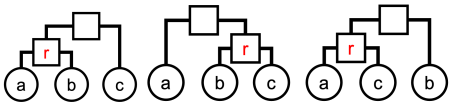
Eq. 8 ensures that the values of $\frac{u}{k}$ that are closer to the original value $\frac{t}{k}$ will have a higher sampling probability, resulting in a closer distance of the original and randomized value of feature i which optimizes the data utility. To satisfy the assigned privacy budget ε_i for feature i, we introduce the noise scale parameter σ_i, bounded to satisfy the guarantee of ε_i-LDP in section IV.

Finally, we randomize each feature by applying FEATURERR on every features independently. The randomized features are concatenated together to create a randomized feature vector $\tilde{z}$, which is used in the training process. The pseudo codes of FEATURERR is presented in Algorithm 1 (Appendix D in our supplemental documents).



(a) Graph $G(\mathcal{V}, \mathcal{E}_{pub} \cup \mathcal{E}_{pri})$ (b) HRG model $(\mathcal{D}, \{p_r, \bar{p}_r\})$

Fig. 4. An instance of HRG model. In (a), the red and black edges are the private and public edges, respectively. In (b), the black and red values are the values of p_r and $\bar{p}_r$ respectively.



(a) $\mathcal{D}(r)$ (b) Two possible versions of $\mathcal{D}'(r)$

Fig. 5. Possible structures of subtree at an internal node r.

B. Edge-aware Randomized Response (EDGERR)

We present our EDGERR mechanism in this section. First, we introduce PHRG model, an alternative to the HRG model. Second, we introduce our MCMC sampling process of PHRG, which leverages the public edges to optimize the utility of the PHRG while providing privacy protection for private edges

a) PHRG model: Let us define a PHRG model (Figure 4) by a tuple $(\mathcal{D}, \{p_r, \bar{p}_r\})$ where $\mathcal{D}$ is a dendrogram with the number of leaf nodes $n = |\mathcal{V}|$, the set of probabilities $\{p_r, \bar{p}_r\}$ is associated with each internal node r in $\mathcal{D}$. In this work, p_r ($\bar{p}_r$) is the probability of having a public (private) edge between the leaf nodes from the left of r and all the leaf nodes from the right of r. We define the likelihood of a dendrogram $\mathcal{D}$ as a product of the likelihood of public edges and the likelihood of private edges as follows:

$$\begin{aligned}\mathrm{L}(\mathcal{D}, \{p_r, \bar{p}_r\}) &= \mathrm{L}_{pub}(\mathcal{D}, \{p_r\})\mathrm{L}_{pri}(\mathcal{D}, \{\bar{p}_r\}) \\ &= \prod_{r\in\mathcal{D}} p_r^{e_r}(1-p_r)^{L_r R_r - e_r} \times \bar{p}_r^{\bar{e}_r}(1-\bar{p}_r)^{\bar{L}_r \bar{R}_r - \bar{e}_r}\end{aligned} \tag{10}$$

where e_r and $\bar{e}_r$ are the public and private edges between the leaf nodes from the left of r and leaf nodes from the right of r; L_r and R_r are the numbers of leaf nodes from the left and the right of r respectively, such that each node has *at least one public edge*; and $\bar{L}_r$ and $\bar{R}_r$ are the numbers of nodes from the left the right child of r respectively, such that each node has *at least one private edge*. In addition, by maximum likelihood, $p_r = \frac{e_r}{L_r R_r}$ and $\bar{p}_r = \frac{\bar{e}_r}{\bar{L}_r \bar{R}_r}$.

To find the most suitable PHRG model representing the graph G, we need to find the optimal dendrogram $\mathcal{D}^*$ that maximize the log-likelihood of Eq. 10, as follows:

$$\begin{aligned}\mathcal{D}^* &= \arg\max_{\mathcal{D}} \big(\mathbb{L}_{pub}(\mathcal{D}) + \mathbb{L}_{pri}(\mathcal{D})\big) \\ &= \arg\max_{\mathcal{D}} \big[-\sum_r N_r \chi(p_r) - \sum_r \bar{N}_r \chi(\bar{p}_r)\big]\end{aligned} \tag{11}$$

where $\mathbb{L}_{pub}(\mathcal{D}) = \sum_r N_r \chi(p_r)$, $\mathbb{L}_{pri}(\mathcal{D}) = \sum_r \bar{N}_r \chi(\bar{p}_r)$, $\{r\}$ is the set of all internal nodes in $\mathcal{D}$, $N_r = L_r R_r$, $\bar{N}_r = \bar{L}_r \bar{R}_r$, and $\chi(\tau) = \tau \log \tau + (1-\tau)\log(1-\tau)$.

b) Edge-level DP MCMC Sampling: It is expensive to find $\mathcal{D}^*$ by generating all $(2n-3)!!$ possible dendrograms. To address this problem, we propose a max-max MCMC process to approximate $\mathcal{D}^*$ while preserving the edge-level DP of $\mathcal{E}_{pri}$. Starting from a random dendrogram $\mathcal{D}_0$, each MCMC sampling step consist of two processes: (1) optimizing the dendrogram using the subgraph $G_{pub}(\mathcal{V}_{pub}, \mathcal{E}_{pub})$ and (2) optimizing the dendrogram using the subgraph $G_{pri}(\mathcal{V}_{pri}, \mathcal{E}_{pri})$ where $\mathcal{V}_{pub}$ ($\mathcal{V}_{pri}$) are the set of nodes in graph G that has at least one public (private) edge.

At a MCMC sampling step t, process (1) randomly samples a dendrogram $\mathcal{D}'$, and updates the current dendrogram $\mathcal{D}_t$ as:

$$\mathcal{D}_t^{(1)} = \begin{cases} \mathcal{D}', & \text{with probability } \eta \\ \mathcal{D}_{t-1}, & \text{with probability } 1-\eta \end{cases} \tag{12}$$

where the acceptance probability $\eta = \min\left(1, \frac{\exp \mathbb{L}_{pub}(\mathcal{D}')}{\exp \mathbb{L}_{pub}(\mathcal{D}_{t-1})}\right)$. To sample $\mathcal{D}'$, we randomly choose an internal node r (not the root) in $\mathcal{D}_{t-1}$ and randomly choose one of the two alternative possible structures of r as $\mathcal{D}'$ (Figure 5).

Similarly, at the (2) process EDGERR randomly samples $\mathcal{D}'$ and updates the current dendrogram $\mathcal{D}_t$ as follows:

$$\mathcal{D}_t = \begin{cases} \mathcal{D}', & \text{with probability } \bar{\eta} \\ \mathcal{D}_t^{(1)}, & \text{with probability } 1-\bar{\eta} \end{cases} \tag{13}$$

where the acceptance probability $\bar{\eta}$ is computed as

$$\bar{\eta} = \min\left(1, \frac{\exp\left(\frac{\varepsilon_{e1}}{\Delta_e}\mathbb{L}_{pri}(\mathcal{D}')\right)}{\exp\left(\frac{\varepsilon_{e1}}{\Delta_e}\mathbb{L}_{pri}(\mathcal{D}_{t'-1})\right)}\right) \tag{14}$$

with Δ_e is the global sensitivity of $\mathbb{L}_{pri}(\cdot)$ bounded in Lemma 4 and a privacy budget ε_{e1}. Since, the MCMC sampling process is reversible and ergodic [15], there exists only one equilibrium state, which assures the convergence condition.

After the sampling process, given $(\mathcal{D}^*, \{p_r, \bar{p}_r\})$, we employ `CalculateNoisyProb` algorithm (Algorithm 3 in Appendix D) [15] to add Laplacian noise to $\{\bar{p}_r\}$ with a privacy budget ε_{e2}. We use the perturbed dendrogram to generate the edge-level DP-preserving subgraph $\tilde{G}_{pri}$, then, we merge $\tilde{G}_{pri}$ with the public graph G_{pub} to construct the (complete) edge-level DP-preserving graph $\tilde{G} = G_{pub} \cup \tilde{G}_{priv}$. The graph $\tilde{G}$ will be used to train the GNNs as the replacement for G to preserve the privacy of private edges $\mathcal{E}_{pri}$. The pseudo codes of EDGERR is presented in Algorithm 2 (Appendix D).

IV. Privacy Guarantees

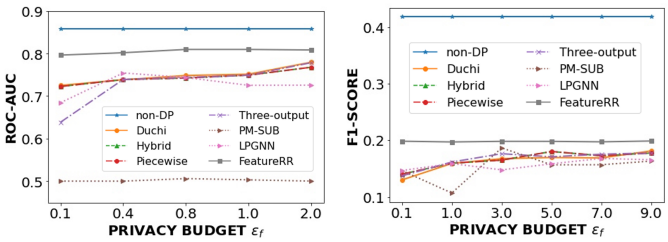
This section analyzes the privacy guarantee of HETERORR at the embedding feature-level LDP and the edge-level DP.

Bounding σ_i in FEATURERR*:* To satisfy ε_f-LDP for the embedding feature z, we need to bound the noise scale σ_i to assure ε-LDP for the embedding feature z.

Theorem 3. *For each feature $i \in [d]$, if $\sigma_i \geq \frac{(k-1)}{k\epsilon\theta_i}$, then our mechanism preserves ϵ-LDP for the whole feature vector x.*

The proof of Theorem 3 is in the Appendix A of our supplemental documents.

Edge-level DP: First, we bound the global sensitivity Δ_e.



(a) FLICKR-MIR (b) PPI

Fig. 6. Model performance of the feature-level protection.

Lemma 4. *Δ_e monotonically increases as $n \to +\infty$, and*

$$\Delta_e = \log N_{max} + \log\left(1 + \frac{1}{N_{max}-1}\right) \tag{15}$$

where $N_{max} = \frac{|\bar{\mathcal{V}}|^2}{4}$ when $|\bar{\mathcal{V}}|$ is even and $N_{max} = \frac{|\bar{\mathcal{V}}|^2-1}{4}$ when $|\bar{\mathcal{V}}|$ is odd with $\bar{\mathcal{V}} \subset \mathcal{V}$ is the set of private nodes.

The proof of Lemma 4 is in the Appendix B of our supplemental documents. Given the bounded sensitivity Δ_e in Lemma 4, our HRG sampling process satisfies edge-level ε_{e1}-DP, since the exponential mechanism has been proved satisfying the DP constraint with a desired privacy budget ε_{e1} [11].

Regarding the `CalculateNoisyProb` algorithm, for two neighboring graphs G and G' that are different at one private edge, the global sensitivity is 1. Thus, perturbing the sampled dendrogram $\mathcal{D}^*$ satisfies ε_{e2}-DP to protect private edges. Following the sequential composition theorem [11], EDGERR satisfies ε_e-DP with $\varepsilon_e = \varepsilon_{e1} + \varepsilon_{e2}$ to protect private edges.

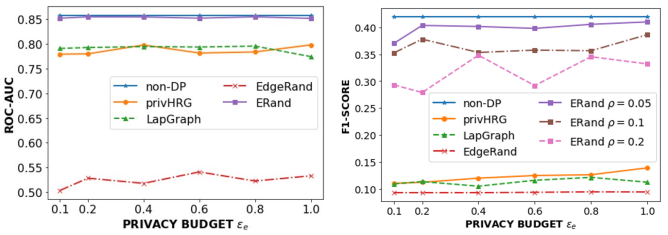
V. Experimental Results

We conduct extensive experiments on benchmark datasets to illustrate interplay between privacy budget and model utility in HETERORR at the feature level, the edge level, and how well it can defend against the PIAs at both levels.

a) Datasets, Model and Metrics: We consider two datasets including FLICKR-MIR and PPI [17]. For the PPI dataset, we select the proportion of private edges $\rho \in \{0.05, 0.1, 0.2\}$ to construct the set of private edges in each graph. For the FLICKR-MIR dataset, we consider the edges constructed by "taken by friends" private ($\rho \approx 0.305$) and others are public edges. We use ResNet-50 [18] with ImageNet weights and RetinaFace [19] to extract embedding features and faces from the images.

We employ the Graph Convolutional Network (GCN) [20] in every experiment. We use an average aggregation function for each layer in the GCN models as in [20]. We use F1-score to evaluate the model performance on the PPI dataset, and we use *ROC-AUC* metric to evaluate the model performance on the FLICKR-MIR dataset. All statistical tests are 2-tail t-tests.

b) Baselines: We consider well-known state-of-the-art LDP mechanism baselines for the feature-level privacy protection, including Duchi mechanism [21], Piecewise mechanism [22], Hybrid mechanism [22], Three-output mechanism [23], Sub-optimal mechanism [23], and (LPGNN) [6]. Regarding the edge-level, we consider three baselines: privHRG [15], EdgeRand [4], and LapGraph [4]. We include the clean model (non-DP) to show the upper bound of the model performance.



(a) FLICKR-MIR (b) PPI

Fig. 7. Model performance of the edge-level protection.

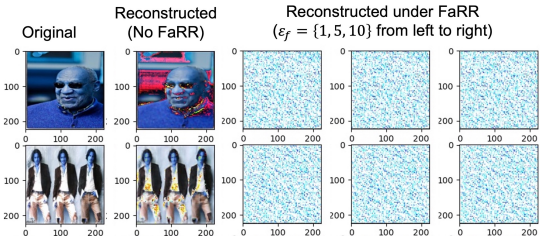


Fig. 8. Results of [24] attack reconstructing the raw images from embedding features extracted from the layer 7th over 176 layers of ResNet-50

c) Results on the Embedding Feature-Level: Figures 6 illustrates the GCN performance associated with the change of ε_f. FEATURERR achieves the best model performance compared to the baselines in all three datasets. For the FLICKR-MIR dataset, FEATURERR has 7.6% improvements on average, respectively, i.e., p-value $= 3.0e-05$, compared with the best baseline (LPGNN). In the PPI dataset, FEATURERR improves the model performance 2.6% compared with the best baseline, i.e., the Three-output mechanism, with p-value $= 0.0031$.

d) Results on the Edge-level: Figures 7 illustrate the results on the FLICKR-MIR, and PII datasets. In the FLICKR-MIR, EDGERR outperforms all the baselines where textscEdgeRR improves 6.3% over the best baseline (LapGraph) with p-value $= 3.6e-09$. In the PPI dataset (Figure 7(b)), we compare the model performance of each algorithm given different values of ρ. We observe that the F1-score of GCN trained is higher when ρ is smaller since the lower ρ reduces the number of private edges being randomized.

e) Defending against PIAs: We conduct the LinkTeller [4] and Image Reconstruction [24] attack to the GNNs under the protection of HETERORR to test its defensive power.

Defending the LinkTeller [4]. We evaluate EDGERR against the LinkTeller attack to analyze its ability in protecting private edges. For the non-DP model, the LinkTeller shows an efficient attack performance (ROC-AUC is 0.91) on the FLICKR-MIR dataset. When we apply EDGERR to protect the private edges, the performance of the LinkTeller is significantly reduced to nearly random guess $[0.52, 0.54]$ given a wide range of the privacy budget $\varepsilon_e \in [0.1, 1.0]$. Therefore, EDGERR is effective in defending the LinkTeller attack.

Defending image reconstruction attack [24]. We train an attacker on the ImageNet dataset to reconstruct the images from embedding features extracted from the pre-trained ResNet-50 model; then, we use the trained attacker to reconstruct the image from the embeddings The results of the reconstructed and original images are in Figure 8. We found that FEATURERR successfully prevent the attacks due to the power of LDP which is consistent with the previous studies.

VI. CONCLUSION

In this paper, we present HETERORR, a mechanism to simultaneously protect nodes' embedding features and private edges under LDP and DP protections in training GNNs. By balancing the sensitivity and importance of features and edges HETERORR retains high data and model utility under the same privacy protection in training GNNs compared with existing baseline approaches. Also, our HETERORR is resistant to PIAs, such as LinkTeller and image reconstruction attacks.

ACKNOWLEDGEMENT

This work is partially supported by grants NSF IIS-2041096, NSF CNS-1935928/1935923, NSF CNS-1850094, and unrestricted gifts from Adobe System Inc.

REFERENCES

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE TNNLS*, 2020.
[2] V. Duddu, A. Boutet, and V. Shejwalkar, "Quantifying privacy leakage in graph embedding," in *EAI MobiQuitous 2020*, 2020.
[3] H. et al., "Node-level membership inference attacks against graph neural networks," *arXiv preprint arXiv:2102.05429*, 2021.
[4] F. Wu, Y. Long, C. Zhang, and B. Li, "Linkteller: Recovering private edges from graph neural networks via influence analysis," *SP 2022*, 2021.
[5] K. Z. L. C. L. S. C. Yang, H. Wang, "Secure deep graph generation with link differential privacy," in *IJCAI 2021*, 8 2021.
[6] S. Sajadmanesh and D. Gatica-Perez, "Locally private graph neural networks," in *CCS'21*, 2021.
[7] S. Kim, H. Shin, C. Baek, S. Kim, and J. Shin, "Learning new words from keystroke data with local differential privacy," *TKDE'20*, 2020.
[8] D. Wang and X. Jinhui, "On sparse linear regression in the local differential privacy model," in *ICML*, 2019.
[9] J. McAuley and J. Leskovec, "Image labeling on a network: using social-network metadata for image classification," in *ECCV*, 2012.
[10] N. Gong, Zhenqiang, and B. Liu, "Attribute inference attacks in online social networks," *ACM TOPS)*, 2018.
[11] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy." *Found. Trends Theor. Comput. Sci.*, 2014.
[12] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith, "Analyzing graphs with node differential privacy," in *Theory of Cryptography*, 2013.
[13] C. Bo, H. Calvin, Y. Kasra, and H. Matthew, "Edge differential privacy for algebraic connectivity of graphs," 2021.
[14] A. Clauset, C. Moore, and M. E. Newman, "Structural inference of hierarchies in networks," in *ICML Workshop*, 2006.
[15] Q. Xiao, R. Chen, and K. Tan, "Differentially private network data release via structural inference," in *KDD'14*, 2014.
[16] S. M. Lundberg and S. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, 2017.
[17] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *NIPS*, 2017.
[18] H. Kaiming, Z. Xiangyu, R. Shaoqing, and S. Jian, "Deep residual learning for image recognition," in *CVPR*, 2016.
[19] J. Deng, J. Guo, E. Ververas, I. Kotsia, and S. Zafeiriou, "Retinaface: Single-shot multi-level face localisation in the wild," in *CVPR*, 2020.
[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *ICLR'17*, 2017.
[21] J. Duchi and R. Rogers, "Lower bounds for locally private estimation via communication complexity," in *Conference on Learning Theory'19*.
[22] W. Ning and others., "Collecting and analyzing multidimensional data with local differential privacy," in *ICDE 2019*, 2019, pp. 638–649.
[23] Z. Yang *et al.*, "Local differential privacy-based federated learning for internet of things," *IEEE IoT Journal*, 2020.
[24] T. B. A. Dosovitskiy, "Inverting visual representations with convolutional networks," *CVPR*, 2016.