Edge Private Graph Neural Networks with Singular Value Perturbation

Tingting Tang*
University of Southern California
Los Angeles, California, USA
tangting@usc.edu

Yue Niu*
University of Southern California
Los Angeles, California, USA
yueniu@usc.edu

Salman Avestimehr
University of Southern California
Los Angeles, California, USA
avestime@usc.edu

Murali Annavaram
University of Southern California
Los Angeles, California, USA
annavara@usc.edu

ABSTRACT

Graph neural networks (GNNs) play a key role in learning representations from graph-structured data and are demonstrated to be useful in many applications. However, the GNN training pipeline has been shown to be vulnerable to node feature leakage and edge extraction attacks. This paper investigates a scenario where an attacker aims to recover private edge information from a trained GNN model. Previous studies have employed differential privacy (DP) to add noise directly to the adjacency matrix or a compact graph representation. The added perturbations cause the graph structure to be substantially morphed, reducing the model utility. We propose a new privacy-preserving GNN training algorithm, `Eclipse`, that maintains good model utility while providing strong privacy protection on edges. `Eclipse` is based on two key observations. First, adjacency matrices in graph structures exhibit low-rank behavior. Thus, `Eclipse` trains GNNs with a low-rank format of the graph via singular values decomposition (SVD), rather than the original graph. Using the low-rank format, `Eclipse` preserves the primary graph topology and removes the remaining residual edges. `Eclipse` adds noise to the low-rank singular values instead of the entire graph, thereby preserving the graph privacy while still maintaining enough of the graph structure to maintain model utility. We theoretically show `Eclipse` provide formal DP guarantee on edges. Experiments on benchmark graph datasets show that `Eclipse` achieves significantly better privacy-utility tradeoff compared to existing privacy-preserving GNN training methods. In particular, under strong privacy constraints ($\epsilon < 4$), `Eclipse` shows significant gains in the model utility by up to 46%. We further demonstrate that `Eclipse` also has better resilience against common edge attacks (e.g., LPA), lowering the attack AUC by up to 5% compared to other state-of-the-art baselines.

KEYWORDS

Graph Neural Networks, Differential Privacy, Singular Value Decomposition, Node Classification, Edge Privacy Attack

* These authors contributed equally to this work

Proceedings on Privacy Enhancing Technologies YYYY(X), 1–16

https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Graph-structured data are ubiquitous in the real world, such as social networks [10], molecules [11], and transportation networks [5]. Graph Neural Networks (GNNs) are a family of powerful neural networks that learn representations from such relational data and have achieved state-of-art performance in learning tasks, mainly node classification [20], link prediction [47] and graph classification [42]. Due to the good performance, GNNs have been deployed in various applications, such as recommender systems [44], drug discovery [27], and traffic flow prediction [35].

Meanwhile, several works have found that GNNs are vulnerable to privacy attacks [14, 40], where edges or node features used in training a GNN can be revealed via interaction with the trained GNN during inference. An attacker can send a series of seemingly innocuous inference requests to a trained model and use the model outputs to infer the structure of the graph used to train the GNN [40], or to extract the node features in the graph [6]. While both the node features and graph structure (often represented by an adjacency matrix) can be sensitive information, in this work we focus on the setting where the node features are public but the graph structure needs to be private. For instance, in a social network a node representation such as the person's education level and city may be public, while that person's social network is private. In general, graph edges usually encode sensitive information, such as private social relationships among users of a social network, molecular structure of a drug under trade secret, or private user-item interaction history of a user in a recommender system [39]. Therefore, designing privacy-preserving GNNs to protect private, sensitive edges against such attacks is critical.

Differential privacy (DP) is a well-known solution to protect against privacy attacks. This technique applies noise to the data while still enabling usage of the perturbed data. In particular, DP has been widely used in machine learning models to protect data and the model. One pioneering work, DP-SGD [1], applies DP-noise to the gradients during the training process. However, the standard DP-SGD algorithm requires gradients of different inputs to be independent. For GNNs with graph data, gradients of each node are affected by its neighboring nodes through node aggregation (See Sec 3.1), making the standard DP-SGD not applicable to GNN settings [2, 3]. Recent works have tailored DP to GNN settings [4, 21, 30, 40]. Specifically, to protect graph edges from being leaked via trained models, several edge-level DP algorithms have been

proposed [21, 30, 40]. For instance, DPGCN [40] directly adds noise to the adjacency matrix and hence the true edge connectivity information is protected. By adding DP noise to the whole adjacency matrix and reconstructing the graph, DPGCN changes the graph structure and reduces the correlation between the true edges in the graph and the graph used during the training process. DPGCN however has lower privacy-utility tradeoff. The reason for the degraded performance is that binary values in the adjacency matrix (1: edge, 0: non-edge) are very sensitive to noise. The perturbed graph structure can be drastically different from the original graph, even for a moderate privacy budget (See more analysis on Sec 3.3). Rather than applying noise to the whole adjacency matrix, LPGNet [21] uses a compact representation of the graph structure called cluster degree vector and applies DP-noise only to this structure. It further tailors the model architecture to exploit the compact graph information. The graph compaction algorithm along with the accompanied modifications to the model architecture enable LPGNet to be more resilient to noise when applying DP, leading to a better model performance with the same privacy constraints. However, the cluster degree vector preserves only coarse-grain information about the graph, namely the number of neighbors in each cluster for a given node, and fully discards information of graph connectivity. Thus the model utility is significantly lower compared to GNN trained with a full adjacency matrix. Furthermore, LPGNet requires a change to the model architecture. In summary, current edge-level DP methods reduce the model utility (F1 score) significantly in favor of privacy and/or require GNN model architecture modifications.

To address the above limitations, we propose `Eclipse`, a privacy-preserving training methodology for training graph neural networks that utilizes the graph structure to improve the model utility. However, our approach applies DP-noise to the low-rank format of a graph, rather than the adjacency matrix of the graph. In particular, the DP-noise is added to the singular values of the matrix obtained via singular value decomposition (SVD). Hence, the dimension to be considered when applying DP is reduced from n^2 to n (n : number of nodes in the graph). Furthermore, as has been observed in prior work [51], adjacency matrices in real-world graph data exhibit low-rank structures. That is, most information in the graph can be compressed into a low-dimensional representation. Hence, the DP-noise addition can be limited to just a few ranks. This key insight allows `Eclipse` to further reduce the dimension that needs to be perturbed by focusing only on a low-rank version of the graph when training GNNs. Unlike LPGNet, `Eclipse` attains the compact format of the graph while still preserving most information, leading to a better privacy-utility tradeoff.

In terms of privacy `Eclipse` protects privacy in two ways. First, with a low-rank graph, `Eclipse` only uses a subset of edges in training, and decorrelates the GNN with the rest of the edges. Then, `Eclipse` further perturbs low-rank singular values and feeds the GNN with a perturbed low-rank reconstruction of the graph during model training, leading to DP on edges in the low-rank graph. Furthermore, `Eclipse` acts as a plug-in module compatible with existing GNN architectures and only needs to replace the original adjacency matrix input with a perturbed low-rank format. Hence, there is no need to change the model architecture.

We empirically evaluate `Eclipse` on 6 datasets under transductive settings (i.e., different nodes from the same graph are used for training and testing), and 1 dataset under inductive settings(i.e., the nodes in the testing graph are unseen during training). In transductive settings, `Eclipse` enjoys much-improved privacy-utility tradeoff over the state-of-the-art baselines, including DPGCN and LPGNet. In particular, under strong privacy constraints ($\epsilon < 4$), `Eclipse` achieves up to 46% higher model utility (F1 score) compared to DPGCN and LPGNet. Even under extreme privacy constraints ($\epsilon < 1$), `Eclipse` still attains up to 5% higher model utility compared to training with node features only (MLP).

We further evaluate `Eclipse`'s resilience against two common edge attacks: LPA [14] and LINKTELLER [40]. Under LPA attacks, `Eclipse` shows better attack resilience compared to DPGCN and LPGNet. Specifically, with comparable model utility among `Eclipse` and other baselines, the trained model using `Eclipse` results in lower Area Under Receiver Operating Curve (AUC), a well known metric for measuring attack performance (lower AUC means better attack resilience). For instance, on the Cora dataset, with a model utility of around 65%, `Eclipse` lowers the attack AUC by 5% compared to LPGNet and DPGCN. Further, attack AUC on `Eclipse` is at most 5% higher compared to MLP, while the attack AUC on LPGNet and DPGCN can go up to 12.5% and 16% higher than MLP, respectively. We also provide insights on privacy protection of `Eclipse` on different parts of the graph, including protection on low-degree and high-degree nodes.

2 RELATED WORK

In this section, we discuss related works on privacy attacks on GNNs, privacy protection using DP, and graph decomposition.

Privacy Attacks on GNNs. Recent studies have demonstrated that GNNs are vulnerable to privacy attacks on different components of a graph, such as nodes, edges, and graph properties (e.g., graph density) [15, 40, 48]. In particular, attacks that recover edges in a graph have been demonstrated. Zhang et al. [49] presents the first model inversion attack on GNNs that reconstructs private edges via white-box access to node attributes, edge density, and the target model parameters. He et al. [14] show black-box link-stealing attacks that can infer the existence of a link between any pair of nodes in the graph with some prior knowledge, such as the target node attributes, extra datasets, etc. LINKTELLER [40] is another important work that proposes a black-box link-stealing attack in a more practical setting based on node influence analysis. And the attack does not require node features when performing attacks. More recently, Olatunji et al. [26] analyze the information leakage based on empirical observation with graph structure reconstruction attacks. They show that additional knowledge of post-hoc feature explanations substantially increases the success rate of these attacks. These works expose the privacy risks of GNNs that use sensitive graph edges during training, and call for effective defense mechanisms to protect the private edges.

Differentially Private GNNs. Differential Privacy (DP) aims to provide formal privacy guarantees for general dataset analysis tasks, and has been widely used in machine learning tasks. It uses a randomized mechanism to perturb the target object and generate indistinguishable outputs for neighboring data. For neural networks, DP-SGD [1] acts as a general privacy-preserving training framework that protects models against model inversion or membership

inference attacks. It injects noise into gradients during training, and therefore reduces correlations between the final model and the input data. However, the standard DP-SGD algorithm requires gradients of different inputs to be independent. For GCNs with graph data, gradients of each node are affected by its neighboring nodes through node aggregation (See Sec 3.1), making the standard DP-SGD not applicable to GCN settings [2, 3]. In the GNN domain, DP is instead tailored to two scenarios: edge-level DP, which protects the existence of an edge [18], and node-level DP, which protects the presence of a node and all the edges connected to it [19].

In the edge-DP setting, Wu et al. [40] propose DPGCN, that directly applies DP on the input graph and uses the perturbed graph to train GNNs. As noise is added to the raw graph data, model performance is affected, particularly under stronger privacy constraints. In contrast to directly perturbing graphs, Kolluri et al. [21] propose LPGNet that adapts a GNN architecture to a low-dimensional graph representation, called *cluster degree vectors*. These vectors are then perturbed via the Laplace mechanism to ensure edge-level privacy. Owing to the reduced dimension when performing DP, LPGNet enjoys better privacy-utility compared to DPGCN. However, model accuracy using LPGNet is capped compared to standard GNNs since limited information is encoded in the degree vectors. Compared to these approaches, our proposed technique attains a compact format of the graph structure and preserves most information about the graph structure at the same time.

In the node-DP setting, Daigavane et al. [4] trains GNNs with node-level DP guarantee by extending the standard DP-SGD algorithm and applying privacy amplification by subsampling on bounded-degree graph data. However, this approach requires large privacy budgets to achieve reasonable model accuracy. Sajadmanesh et al. [30] consider both node-level and edge-level DP, and propose a new GNN architecture GAP to protect both node features and edges. In GAP, node embedding aggregation is decoupled from the standard GNN architecture. They apply noise to the output of the node embedding aggregation (with the adjacency matrix), therefore achieving both edge-level and node-level DP.

There are alternative approaches to protect data privacy during training and inference, such as Secure Multi-Party Computing (MPC) [36–38], Coded Computing [33, 45], and Trusted Execution Environment (TEE) [13, 22, 34]. However, these approaches target different settings from ours, and cannot protect against edge attacks exploiting correlations among model outputs. Thus, in this work, we do not consider these approaches and instead focus on the DP-based approach to provide edge privacy.

Applying SVD on GNNs. Singular value decomposition (SVD) is a commonly used technique in data processing. By encoding the most important information into a low-dimensional space, it shows significant benefits in reducing complexity in DNNs [16, 17, 23–25]. Recent endeavors have also shown its potential in graph tasks, especially for defending against adversarial attacks on graph data. Entezari et al. [9] demonstrated that small perturbation by an adversarial attack is only reflected on high-rank residual components, whereas the low-rank approximation is minimally affected. They exploit this observation to enable GNN training with a low-rank graph, which is more immune to adversarial attacks. Xu et al. [41] aim to speed up robust structure learning, which is another method to improve the robustness of GNNs against adversarial attacks. Their approach exploits the low-rank property of the adjacency matrix of a graph as prior knowledge, and decouples the adjacency matrix into a low-rank component and a sparse one, and learns by minimizing matrix rank. Despite the promising applications of SVD demonstrated in improving the robustness of GNNs, it is rarely seen employing SVD in the privacy domain when training GNNs. In this work, we aim to use SVD to study the graph structure and exploit its potential in privacy-preserving GCN training.

3 PRELIMINARIES

3.1 Graph Neural Networks

GNNs learn a node embedding for each node by distilling the high-dimensional information about a node's graph neighborhood into a low-dimensional embedding. The learned node embeddings can then aid downstream graph learning tasks, such as node classification, link prediction, and clustering. In this work, we focus on graph convolutional neural network (GCN), a prominent GNN type widely used for semi-supervised graph classification tasks on graphs. Consider an unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes ($n = |\mathcal{V}|$), and $\mathcal{E}$ the set of edges which can be represented by an adjacency matrix $A \in \{0,1\}^{n\times n}$, with $A_{ij} = 1$ indicating the presence of an edge between node i and node j. GCN takes as input the adjacency matrix A and the node feature matrix $X \in \mathbb{R}^{N\times d}$, where each row of X corresponds to a d-dimensional feature vector of a node in $\mathcal{V}$. A k-layer GCN consists of a sequence of k graph convolutional layers. To obtain a new embedding for each node in the graph, each hidden layer aggregates the embeddings of the node's neighbors output by the previous layer. The l-th layer can be formally described as:

$$H^{l+1} = \sigma(\tilde{A}H^l W^l), \tag{1}$$

where $\tilde{A}$ is normalized adjacency matrix, σ is the nonlinear activation function such as ReLU, H^l is the node embeddings in the l-th layer where $H^0 = X$ is the initial node feature, and W^l is the learnable weight parameters. $\tilde{A}H^l$ denotes the feature aggregation step. The normalized $\tilde{A}$ can be calculated as

$$\text{FirstOrderGCN:} \quad \tilde{A} = I + D^{-0.5} \cdot A \cdot D^{-0.5}, \tag{2}$$

or

$$\text{AugNormAdj:} \quad \tilde{A} = (D + I)^{-0.5} \cdot A \cdot (D + I)^{-0.5}, \tag{3}$$

where $D = \texttt{Diag}(d_0, d_1, \cdots, d_{|V|-1}), d_i$ denotes degree of i-th node. Depending on datasets and models, different methods may prefer different normalization methods.

Node embeddings from the last layer are transformed into probability scores via a softmax layer for the node classification task. A typical 2-layer GCN is illustrated in Figure 1.

Node-level classification can occur in two settings: *transductive and inductive* settings. In the *transductive setting*, the training and testing are performed on the same graph but with different nodes. While in the *inductive setting*, the graph in testing is distinct from the one in training.

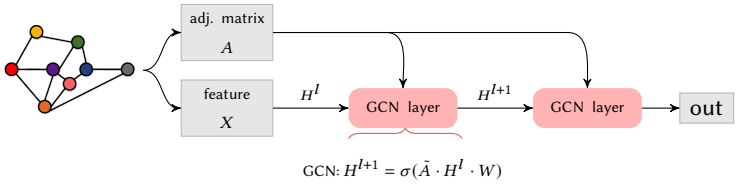

Figure 1: A typical 2-layer GCN. Each GCN layer takes as input node embedding, H^l, and performs aggregation $\tilde{A} \cdot H^l$ with normalized adjacency matrix $\tilde{A}$. Then, a weight matrix W is applied to the aggregated embeddings.

3.2 Differential Privacy

Differential privacy (DP) has been known as a rigorous privacy-preserving methodology and has been widely adopted in privacy-sensitive applications [7].

Definition 1. *(ϵ, δ)-DP: A randomized mechanism $\mathcal{M} : \mathbb{D} \to \mathbb{R}$ satisfies (ϵ, δ)-DP if for any two adjacent inputs $x, x' \in \mathbb{D}$ and any output $\mathbb{S} \subseteq \mathbb{R}$, it holds that*

$$P[\mathcal{M}(x) \in \mathbb{S}] \leq e^{\epsilon} \cdot P[\mathcal{M}(x') \in \mathbb{S}] + \delta.$$

Depending on the applications, the concept of adjacent data varies. In our case, we target adjacency matrix protection of undirected unweighted graphs in graph neural networks. We define adjacent matrices A, A' if *they only differ in one edge*. That is, the graph represented by A' is obtained by adding/removing one edge in the graph represented by A. Specifically, if two graphs differ on one edge between node i and node j, for the *symmetric* adjacency matrices A and A', they differ at (i, j) and (j, i) (i.e., $A(i, j) = A(j, i) = 1$, $A'(i, j) = A'(j, i) = 0$). By replacing x with A in Def 1, we define edge-level DP as follows.

Definition 2. *(ϵ, δ)-edge DP: A randomized mechanism $\mathcal{M}$ satisfies (ϵ, δ)-edge DP if for any two adjacent graphs A, A' and any output $\mathbb{S} \subseteq \mathbb{R}$(range of $\mathcal{M}$), it holds that*

$$P[\mathcal{M}(A) \in \mathbb{S}] \leq e^{\epsilon} \cdot P[\mathcal{M}(A') \in \mathbb{S}] + \delta.$$

However, DP often sacrifices performance for strong privacy constraints (small ϵ). Balancing model performance and privacy of the adjacency matrix is a challenge when working with GNNs.

3.3 Edge-Level DP on GNN

In this section, we describe two typical approaches to edge-level DP on GNN: DPGCN and LPGNet.

DPGCN [40] directly adds noise to the adjacency matrix A and reconstructs the graph as

$$\hat{A} = \texttt{binary}(A + \texttt{Laplacian}(0, \sigma)),$$

where σ controls the variance of a Laplacian noise matrix that has the same dimension as A. The noise added is determined by the ϵ value selected. The smaller the ϵ larger is the noise. The `binary` op simply sorts values in A and sets the first $|\mathcal{E}|$ (number of edges in A) to 1 and zeros to other values. Therefore, the perturbed graph $\hat{A}$ still has similar sparsity as the original A, but with different connections. Since A consists only of binary values, the resulting $\hat{A}$ can be drastically different with a certain level of noise added. That is the main reason that DPGCN's performance quickly drops when the privacy budget decreases.

LPGNet mitigates the issue by extracting a much-compressed representation of the graph called cluster degree vectors, D. Specifically, for each node, LPGNet computes the number of its neighbors on each cluster (i.e., nodes with the same label), and perturbs the degree vectors as

$$\hat{D} = D + \texttt{Laplacian}(0, \sigma),$$

where each row in $D \in \mathbb{Z}^{n \times L}$ denotes the number of the node's neighbors in each cluster, L indicates the number of clusters/labels, σ controls the variance of the Laplacian noise matrix that has the same dimension as D. By using the compressed representation, LPGNet discards detailed connection information and sacrifices model performance with an improved privacy guarantee.

4 PROBLEM SETTINGS AND THREAT MODEL

Problem Settings. We target a node classification problem over a graph when node features are public and edges are private. The problem arises in real scenarios where edges may represent sensitive social or financial relationships. Such relational data can significantly enhance GNNs to learn representations and relations from public node features [21, 50]. For instance, social networks such as LinkedIn and Twitter allow users to hide their connections for strong privacy protection (edges), while still maintaining a public profile (node features). In these scenarios, private connections may contain sensitive information not present in node features.

The entity with node features trains GNNs with additional information from the graph provider. After training, the model is deployed, and receives queries for classifying certain nodes (e.g., identifying user groups). However, even if the GNN model is trained using a private adjacency matrix, it is possible that the adjacency information can be leaked by interacting with the model. For instance, well-known attack methods such as LPA [14] and LINKTELLER [40] can effectively estimate the graph edges by sending a series of inference queries and analyzing the model's outputs.

Therefore, this work aims to *train a GCN model with good utility using private graph data, while still maintaining strong protection on the graph during/after training.*

Threat Model. We assume an adversary has access to node features and models' API. It learns certain prior knowledge about a target graph through node features. It can also query models with node features with an unlimited number of queries. However, the adversary has no access to the private graph structure A.

5 PROPOSED METHOD

Based on the data shown in prior works [40], it is best to avoid applying DP noise to the adjacency matrix A, as the noise alters the

graph structure significantly leading to performance drop. On the other hand ignoring the graph structure entirely is also unacceptable. In fact, prior works have demonstrated that using a distilled version of a graph [21] improves model performance. However, the performance gap is still high compared to a non-private GCN training approach. The reason is that very limited information is preserved when converting the graph into a compact format as in [21]. Based on these observations this work presents a new approach called `Eclipse` that trains a GCN with a low-rank format of a graph. The low-rank format preserves as much information about a graph as possible but with a low-dimensional format.

We first demonstrate that the difference between A and A', measured as l_2 norm, can be transferred to their singular values, therefore reducing the dimension of DP-noise from n^2 to n. Thus the amount of noise added for a given ϵ privacy budget can be reduced. Furthermore, we observe that high-rank singular values differ more compared to the low-rank part. With the observation, `Eclipse` trains GCNs with a low-rank approximation of A. To achieve DP, `Eclipse` only needs to add noise to the low-rank singular values. Owing to the reduced dimension, `Eclipse` achieves high accuracy while maintaining strong privacy protection.

5.1 Graph Decomposition

In this section, we describe the graph decomposition that we use in our scheme. The essence of the graph decomposition in `Eclipse` is to reduce dimensions for applying DP noise, therefore leading to better privacy-utility trade-offs when applying DP. We then demonstrate how the l2 norm difference between A and A' adjacency matrices is reflected in their singular values.

Given graph $A \in \{0,1\}^{n\times n}$ and $A' \in \{0,1\}^{n\times n}$ (with one edge removed from A), we first decompose A as

$$A \xrightarrow{SVD} U \cdot \texttt{Diag}(s) \cdot V, \tag{4}$$

where $S = \texttt{Diag}(s)$ maps singular values s to a 2-D diagonal matrix, U, V denotes the principal components of A. In general, for a matrix, SVD finds a set of orthogonal basis vectors, U, V, to represent the matrix. The singular values obtained from SVD reflect the amount of information of the matrix contained along corresponding basis vectors. With this notion, we apply SVD to the adjacency matrix A, and analyze its bases and each basis vector's importance.

Decomposition on A'. If performing a normal SVD on A' as in Eq(4), it ends with three matrices that may be different from the ones obtained from A. Hence applying DP to protect graph connectivity requires adding noise to all the three corresponding matrices, U, V, and S, which goes against our objective of applying noise to a reduced dimension graph information.

One important observation is that given A' and A only differ at one edge, their principal bases (i.e., U, V) after the SVD operation tend to be similar. Specifically, we randomly remove one edge in A, and compute the cosine similarity of every principal basis between A and A' as

$$\texttt{sim}_U = \frac{\langle U_i, U_i' \rangle}{\|U_i\| \cdot \|U_i'\|}, \quad \texttt{sim}_V = \frac{\langle V_i, V_i' \rangle}{\|U_i\| \cdot \|U_i'\|},$$

where U', V' are the principal bases from A' via normal SVD, $U_i = U(:, i), V_i = V(:, i)$, so as U_i', V_i'. Large similarity indicates two vectors are close. For a graph A, we denote a A' the worst case if it has the smallest average cosine similarity to A across all principal components. For instance, Figure 2 shows the cosine similarity of the top 20 principal basis vectors on the Chameleon dataset [31]. Owing to the closeness of A and A', their principal components have cosine similarity close to 1. The difference between A and A' is mainly on their singular values. More empirical evidence on other datasets is shown in Appendix B. This observation motivates us to apply DP only to the singular values, rather than n^2 elements in the original graph, and leave the principal basis untouched. Our privacy analysis in Section 6.2.2 discusses the implication of privacy protection beyond this assumption.

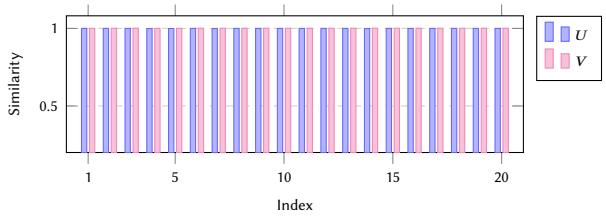

Figure 2: Cosine similarity of principal basis vectors in the Chameleon dataset.

Therefore, we can let A' be *anchored* to the principal basis of A as

$$A' \rightarrow U \cdot S' \cdot V \tag{5}$$

The essence of Eq(5) is to show that A and A' can share the same basis, while their difference is reflected only in their singular value matrices via proper decomposition. Then we can obtain S' as

$$S' = U^T \cdot A' \cdot V^T. \tag{6}$$

Low-Rank Structure. We further observe that graph matrix A exhibits a low-rank structure, with singular values decaying quickly at the beginning as shown in Figure 3. The low-rank structure is rooted in real-world observations. For instance, in social networks, users close to each other tend to have similar connections, leading to highly correlated vectors in the adjacency matrix.

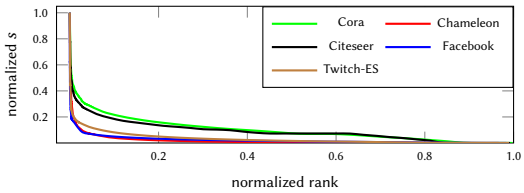

Figure 3: Distribution of singular values (normalized) in typical graph data. Singular values decay quickly, indicating that the graph matrix exhibits a highly low-rank structure.

With such an observation, we can approximate graph A with a low-rank version A_{lr} as

$$A_{\text{lr}} = U(:, 1:r) \cdot \texttt{Diag}(s(1:r)) \cdot V(1:r, :), \tag{7}$$

where $U(:, 1:r), V(1:r, :)$ denotes the first r columns and rows in U, V.

According to Eq(5), we can also write the low-rank version of A' as

$$A'_{\text{lr}} = U(:, 1:r) \cdot \texttt{Diag}(s'(1:r)) \cdot V(1:r, :), \tag{8}$$

with the same basis as in A, s' denote the diagonal elements in S'.

Approximating the graph with a low-rank format is beneficial in two aspects. First, with a low-rank graph, many edges from the original graph are removed. Therefore, training GCNs with the low-rank graph de-correlates the final model from these edges. Second, with the low-rank approximation, we further reduce the dimension to be considered and only need to perturb the first r singular values when applying DP.

5.2 DP on Singular Values

With the notation that A and A' can share principal bases while differing in their singular values, we use a ℓ_2-sensitivity to define the difference between A and A' as

$$\Delta = \max_{A,A'} \left\| s(1:r) - s'(1:r) \right\|_F \tag{9}$$

With the ℓ_2-sensitivity, we apply a standard Gaussian mechanism to perturb the singular values as follows:

Gaussian Mechanism on Singular Values. For a low-rank approximation A_{lr} with r bases, a noise vector $g \in \mathbb{R}^r$ following normal distribution is generated as $g \sim \texttt{Normal}(0, \sigma)$. Then the noise vector g is added to the singular values as $\tilde{s} = s + g$.

The noise parameter σ depends on privacy constraint ϵ and sensitivity Δ. The following lemma shows the sensitivity, Δ, is well-bounded given A and A' share the same principal bases.

Lemma 1. *Let $s = \texttt{Diag}(U^T \cdot A \cdot V^T)$ and $s' = \texttt{Diag}(U^T \cdot A' \cdot V^T)$, where U and V are principal bases obtained from SVD. Given A and A' share principal bases, we have $\Delta \le \sqrt{2}$.*

Proof. Knowing that A and A' differ in one edge,

$$\left\| A - A' \right\|_2 = \sqrt{2}.$$

The goal is to bound the following quantity:

$$\left\| s(1:r) - s'(1:r) \right\|_F.$$

Given A and A' share principal bases U, V, and U, V are unitary matrices,

$$\begin{aligned}
\left\| s(1:r) - s'(1:r) \right\|_F &= \sqrt{\sum_{i=1}^{r} (s(i) - s'(i))^2} \\
&\le \sqrt{\sum_{i=1}^{n} (s(i) - s'(i))^2} = \left\| s - s' \right\|_F \\
&= \left\| U \cdot \texttt{Diag}(s - s') \cdot V \right\|_F = \left\| A - A' \right\|_F \\
&= \sqrt{2},
\end{aligned}$$

which concludes the proof. □

Then we can compute σ given a privacy budget ϵ (See Privacy Analysis in Sec 6). A high privacy constraint (small ϵ) or higher sensitivity requires large noise (high σ) for singular values.

5.3 GCN Training and Testing

`Eclipse` follows the same procedure as standard GCNs, except that a perturbed low-rank adjacency matrix is fed to the model. As shown in Figure 4, during GCN training, we first obtain a perturbed low-rank format, A_{lr}. Since the adjacency matrix is used as an input and does not change during training, we only need to perturb it once and fix it throughout the whole training period. To avoid additional overhead when performing SVD, we compute A_{lr} offline so that it does not affect the training time.

Considering the original adjacency matrix A only consists of binary values, we further align A_{lr} with the standard format by binary quantization. Specifically, given the number of edges, $|\mathcal{E}|$, in the original graph, we sort elements in A_{lr} and set the first $|\mathcal{E}|$ largest values to 1, and zero other values in order to preserve the sparsity of the adjacency matrix. The resulting adjacency matrix is denoted as A_{plr}. In addition, we further consider perturbing the number of total edges E. To that end, we distribute the privacy budget into ϵ_e and ϵ_{lr}, where ϵ_e is for the number of edges, and ϵ_{lr} is for the low-rank adjacency matrix. By the composition rule of DP, the total privacy budget on the graph edges is still ϵ. Algorithm 1 summarizes the perturbation mechanism in our method `Eclipse`.

Algorithm 1 Perturbation mechanism in `Eclipse`

Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as symmetric adjacency matrix A, rank r, privacy parameters ϵ, δ, sensitivity $\Delta = \sqrt{2}$, randomization generator

Ensure: the perturbed low rank outcome A_{plr}

$\epsilon_e = 0.01\epsilon$
// Distribute privacy budget
$\epsilon_{\text{lr}} = \epsilon - \epsilon_e$
$E = |\mathcal{E}|$
$\tilde{E} = E + \texttt{Laplacian}(0, \frac{1}{\epsilon_e})$
$\tilde{E} = \lfloor \tilde{E} \rfloor$
$U, \texttt{Diag}(s), V = SVD(A)$
for $i = 1, \cdots, r$ **do**
  $\tilde{s}(i) = \tilde{s}(i) + \texttt{Normal}(0, \frac{\Delta\sqrt{2\ln(1.25/\delta)}}{\epsilon_{\text{lr}}})$
end for
$A_{\text{lr}} = U(:, 1:r) \cdot \texttt{Diag}(\tilde{s}(1:r)) \cdot V(1:r, :)$
$A_{\text{tr}} = \texttt{UppTriMatrix}(A_{\text{lr}}, 1)$
// Upper triangular matrix on 1^{st} superdiagonal
$top\_indices = \text{argmax}(A_{\text{tr}}, \tilde{E})$
// Get the indices of top $\tilde{E}$ values
$A_{\text{tr}}[top\_indices] = 1, A_{\text{tr}}[\neg top\_indices] = 0$
$A_{\text{plr}} = A_{\text{tr}} + A_{\text{tr}}^T$
return A_{plr}

Transductive Settings. In a transductive setting, as the same graph is used in training and testing, we only need to apply the SVD and DP offline process once and obtain A_{lr} for both training and testing. That is, given nodes chosen in training and testing, we extract the edge information from A_{lr} that covers the chosen nodes.

Inductive Settings. In an inductive setting, we apply the SVD and DP offline process separately on the training and testing graph and obtain two perturbed low-rank graphs.

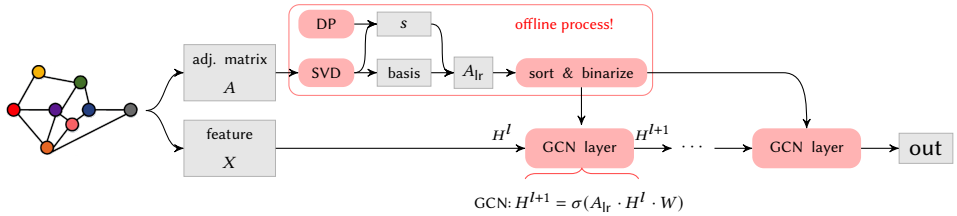

Figure 4: GCN with low-rank graph. Eclipse first applies SVD to obtain singular values of a graph, and perturbs singular values with Gaussian noise. Eclipse then reconstructs a low-rank version of A by only keeping most principal components. To align with original format of the adjacency matrix, Eclipse applies a binary quantization before feeding A_{lr} to a GCN.

6 PRIVACY ANALYSIS

In this section, we analyze privacy protection using Eclipse from two aspects: formal privacy analysis with DP and additional privacy protection via low-rank decomposition. We further discuss the effectiveness of Eclipse in real-world scenarios.

6.1 Formal DP Analysis

Based on the observations on real-world datasets (e.g. Figure 2), it is reasonable to assume that two adjacent graphs share the same principal bases. The formal privacy analysis is built based on this assumption. We also discuss the implication of privacy protection beyond this assumption in Sec 6.2.

Given an adjacency matrix A, Eclipse decomposes A into principal bases U, V, and singular values s (See Eq(4)). As the principal bases are shared with its neighboring graph A', Eclipse achieves $(\epsilon, \delta)-$ edge DP by only perturbing singular values.

Theorem 1. *Given A and A' share the same principal bases, with Gaussian noise,*

$$\texttt{Normal}\left(0, \Delta\sqrt{2\ln(1.25/\delta)}/\epsilon_{lr}\right),$$

added to singular values; and Laplacian noise,

$$\texttt{Laplacian}\,(0, 1/\epsilon_e),$$

added to the edge count, the perturbed low-rank A_{plr} using Eclipse *satisfies (ϵ, δ)-edge DP with $\epsilon = \epsilon_{lr} + \epsilon_e$.*

The proof follows the standard Gaussian mechanism, the composition theorem, and the post-processing rule of DP algorithm [8]. The detailed proof is given in Appendix A.

Corollary 1. *Given node features and the adjacency matrix with* independent *distributions, the GCN model trained by Algorithm 1 is (ϵ, δ)-edge DP if the perturbation mechanism in* Eclipse *is (ϵ, δ)-edge DP.*

Proof. Given that the Eclipse ensure (ϵ, δ)-edge DP, DP guarantee of the trained GCN model with the perturbed adjacency matrix directly follows the post-processing rule [8].

As we assume node features and the adjacency matrix have independent distributions, node features will not contain the adjacency information. According to the post-processing rule, any function with the adjacency matrix as input will not increase privacy leakage. Therefore, by regarding the trained GCN model as a function, output from the model still ensure (ϵ, δ)-edge DP. □

Remark 1. ***Inference Privacy.*** *In transductive settings, the model is fed with a sub-graph from the same graph as in the training stage. Therefore, Algorithm 1 is (ϵ, δ)-edge DP during inference. In inductive settings, as the model is fed with a graph different from the one in the training stage,* Eclipse *applies the same perturbation as in Algorithm 1. Therefore,* Eclipse *still ensures (ϵ, δ)-edge DP.*

6.2 Privacy Protection in Practice

6.2.1 Privacy Protection via Low-Rank Decomposition. Besides the DP guarantee by perturbing singular values, Eclipse provides additional empirical protection via low-rank decomposition. As stated in Sec 5.1, Eclipse trains GNNs using a low-rank graph, A_{lr}, which only consists of the first r principal components in the original graph. As a result, A_{lr} only consists of the primary topology, while discarding the rest of the connections (See more empirical studies in Sec 7.3.2). Owing to such a low-rank decomposition, it is intuitive to observe that Eclipse protects connections that are not included in the low-rank graph, A_{lr}. Meanwhile, Eclipse leverages the primary topology in A_{lr} to train GNNs.

6.2.2 Privacy Protection beyond Assumption. The establishment of Theorem 1 relies on the assumption that A and A' share principal bases. In practice, the effectiveness of the DP guarantee is conditioned by how closely the assumption aligns with the characteristics of real datasets.

To determine if a graph aligns with the assumption, we can get the singular value matrix, S', from A' using the projection in Eq(6), and compare the magnitude of off-diagonal values in S' with its diagonal values. We can further adopt the Monte Carlo method to randomly delete/add one edge in A for better computation efficiency. In the worst case among all A', if the diagonal values are larger than the off-diagonals by several orders of magnitude, the graph and its adjacent will share very similar principal bases. Eclipse will provide strong privacy protection by using top r principal bases and perturbing the corresponding singular values.

The assumption may fail in extreme cases (See Appendix D for one extreme example of this case). In such a scenario, the DP guarantee may not hold. For real-world datasets used in this paper, such extreme cases, that adding/deleting one edge results in completely different principal bases, were non-existent.

Remark 2. *Prior work [32] provides theoretical bounds on the difference in singular values and principal bases (Theorem 2 and 4) given small perturbation on a matrix. In* Eclipse*, adjacency matrices only differ at one edge. Hence, the actual difference in singular values and principal bases is within the bound established in [32]. We leave theoretical justification on the assumption of invariance of principal bases for future work.*

Table 1: Overview of dataset statistics.

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Chameleon	2,277	36,101	2,325	5
PubMed	19,717	44,338	500	3
Facebook	22,470	171,002	128	4

‘

Dataset	Nodes	Edges	Features	Classes
Twitch-ES	4,648	59,382	3,170	2
Twitch-RU	4,385	37,304	3,170	2
Twitch-DE	9,498	153,138	3,170	2
Twitch-FR	6,549	112,666	3,170	2
Twitch-ENGB	7,126	35,324	3,170	2
Twitch-PTBR	1,912	31,299	3,170	2

7 EMPIRICAL EVALUATION

7.1 Evaluation Setup

Datasets. We use 6 standard datasets for the node classification task used in prior work [20, 29, 40]. In the transductive setting, we use Cora, Citeseer, Pubmed [31], Chameleon, and Facebook [29] datasets. Facebook is a social network where nodes represent Facebook pages (sites) and the links represent mutual likes between sites. Node features are extracted from the page descriptions provided by the site owners. Cora, Citeseer, Pubmed, and Chameleon are citation networks where the nodes are documents and the edges are citations. The features represent the existence/non-existence of certain keywords. In the inductive setting, we use Twitch dataset [29] which is a collection of 6 disjoint social networks. In Twitch dataset, nodes represent Twitch streamers from different countries and edges represent their mutual friendships. The features for each streamer are based on the games played and streaming habits. Across all graphs, the node features have the same dimension and semantic meaning. The task is to classify whether a node (streamer) uses explicit language. We train GNNs on the graph corresponding to Spain and use other graphs for testing as done previously. Table 1 summarizes the statistics of the datasets. We follow the same data splits for training for Cora, Citeseer, Chameleon, and Facebook as considered by the previous works [20, 43]. For Cora, Citeseer, and Facebook, the training set consists of randomly chosen 20 labels per class, the test set consists of 1000 nodes and the validation set consists of 500 labeled examples. Chameleon comes with 10 random splits of the nodes with 48% of them used for training, 32% for validation, and 20% for testing. The entire graph structure is used for training. For Twitch we do not use any validation set, instead, we use the models with the lowest training loss.

Model Architectures. We have 4 model types for all our experiments: GCN (without any privacy as a roofline for model performance), DPGCN, MLP, LPGNet, and `Eclipse`. For GCN, we adopt the standard architecture as described in Section 3.1 along with ReLU activation and dropout for regularization in every hidden layer. For MLP, we use a fully-connected neural network with ReLU activation and dropout in every hidden layer. MLP is by design an edge privacy preserving model since MLP does not use adjacency matrix in the training process and instead relies purely on learning from node features. For LPGNet, we use the same model as in LPGNet [40] and choose the LPGNet-2 variant which uses 2 hidden layers and shows higher model accuracy; dropout rate and size of the hidden layers are hyperparameters that can be tuned. For `Eclipse`, we adopt the same architecture as GCN with the adjacency matrix perturbed according to procedures as described in Section 5.3. Specifically, we set rank equal to 20, and further investigate the effect of rank in Section 7.4. We also compare `Eclipse` with the most recent differentially private graph synthesis method PrivGraph, and the results are deferred to Appendix C. We find the right hyperparameters by performing a grid search over a set of possible values in the non-DP setting. We choose the following values for hyperparameter tuning for all models.

- Learning rate: [0.001, 0.005, 0.01, 0.05]
- Hidden layer size: [16, 64, 256]
- # Hidden layers: [2, 3] for GCN, `Eclipse` and all MLPs.
- Dropout: [0.1, 0.3, 0.5, 0.8]

Further, we use two kinds of normalization methods of adjacency matrices for GCN. For Twitch, we use the First Order GCN normalization technique similar to previous works [28](See Eq(2)). For all other datasets, we use the Augmented Normalized Adjacency technique (See Eq(3)).

In the DP setting, we evaluate using the optimal hyper-parameters obtained from the non-DP setting.

Training and Evaluation Details. We adopt cross-entropy loss and optimize using Adam's optimizer with weight decay. For all datasets (except Twitch), we train the models, for 500 epochs and pick the checkpoint that gives the best performance on the validation set during training. This helps us measure the privacy leakage for the model with the best utility. For Twitch, we follow the same training procedure as in LINKTELLER paper[40] and LPGNet paper [21] with 200 epochs. We evaluate the node classification performance using the micro-averaged F1 score following previous works [12, 46] for all datasets except Twitch. During inference, node features of a set of test nodes and the adjacency matrix of the test graph (which is the same as the training graph in transductive setting, or different from the training graph in inductive setting) are input into the trained model. The model then outputs the prediction probability vector and selects the class with the highest probability as the predicted label for each test node. The accuracy (F1 score) is computed by comparing the predicted label against the ground-truth label of the test nodes. For the Twitch dataset, as proposed by the LINKTELLER paper, a modified F1 score is used, which computes the F1 score for the rare class due to significant class imbalance in the dataset. As the task on Twitch dataset is binary classification, we identify the class with fewer data samples as the rare class and consider the rare class as the positive class when calculating the F1 score. We run the training for 5 seeds for all models and report the averaged results.

Attack Overview. We evaluate `Eclipse`'s resilience against two attacks: LINKTELLER [40] and LPA [14]. LINKTELLER establishes that predictions of two connected nodes are more easily influenced by changes in one node's features, compared to the nodes that are

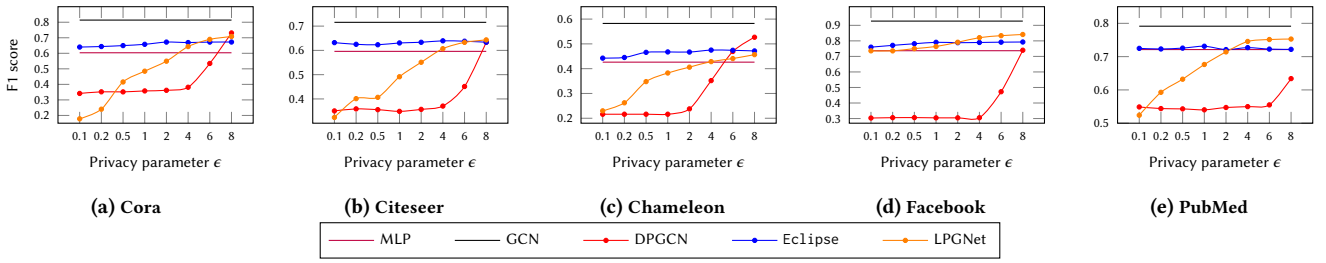

Figure 5: Model utility in transductive setting for various ϵ. `Eclipse` achieves much better model utility under high privacy constraints ($\epsilon < 6$) compared to DPGCN and LPGNet. Even under strong privacy constraints with $\epsilon < 1$, `Eclipse` still achieves better performance compared to MLP without using adjacency matrices during training.

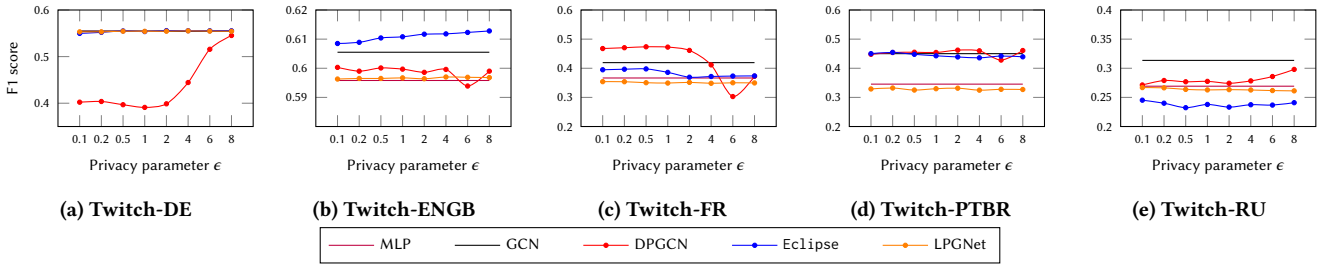

Figure 6: Model utility in inductive setting for various ϵ. `Eclipse` gives consistent performance under different privacy budgets. In contrast, on the Twitch-DE dataset[1], DPGCN's performance collapses under strong privacy constraints. LPGNet also shows poor performance in inductive setting, even under moderate privacy constraints.

not connected. The attacker has access to a set of node features and their labels which are required during training. The attacker's capability also includes the query access to a blackbox GNN model and the obtained prediction probability for a set of nodes during inference. The attacker then computes the *influence* by perturbing the features of a target node and observing the changes in the prediction probability vector of all other nodes from the GNN's outputs. With the influence on all nodes, LINKTELLER can reveal the exact edges of the graph used during inference. As a result, GCNs and other GNN architectures that use graph edges as input are vulnerable to LINKTELLER attacks, that capture the influence propagated through the edges of the graph for such GCNs.

Another attack we consider is LPA. LPA is a more powerful attack that makes use of node features and node embedding correlations to conduct the attack [14] and infer the target graph structure. It is based on the assumption that nodes with more similar features tend to be connected, which is true for homophily graphs. LPA has several different types of attacks based on the knowledge accessible to the attackers, such as node attributes, partial graphs, and a shadow dataset. In our settings where node features are public while the graph structure is completely private, we adopt the LPA attack mode that infers node connections solely based on node features. Specifically, the LPA attack queries the target model with the known node features and computes the distance between posteriors (i.e., model output probability vectors). Detailed distance metrics can be found in Table 13 in [14]. A smaller distance between two nodes' posteriors indicates a higher likelihood of having an edge between them. We follow the protocols of both attacks and reproduce them in our experiment.

Attack Settings. We follow the same attack procedures set by LPA and LINKTELLER. For transductive settings, we evaluate the attacks by how well they can classify existing edges and non-edges from a randomly sampled sub-graph. Specifically, we construct a balanced set by sampling an equal number of edges and non-edges (500 in our experiments). For inductive settings, we sample a set of 500 inference nodes that are not seen previously during training, and create a subgraph. We then evaluate the attacks on all the edges and non-edges in the subgraph. Note that the number of edges and non-edges can be different in the inductive setting. To understand their attack on node degree distribution, as done in LINKTELLER, we also break down the attack performance into low-degree and high-degree nodes and report the results in Section 7.3.

We use Area Under Receiver Operating Curve (AUC) to measure the attack performance, which is used in LPA and LINKTELLER works. For graph-structured data, the current methods measure AUC by how well an attack can separate a randomly sampled edge from a non-edge. Higher AUC indicates that the attack has a higher success rate of identifying the edges used by the model, or equivalently, the model has lower resilience to the attack. We report the average performance of each attack across 5 runs with different seeds.

7.2 Privacy-Utility Tradeoff of `Eclipse`

We first show the privacy-utility tradeoff of `Eclipse` in transductive and inductive settings. We train GCN models with different privacy budgets ϵ, ranging from 0.1 to 8. Small ϵ indicates strong privacy constraints. Besides the baseline methods (i.e., DPGCN, LPGNet),

[1] On Twitch-DE, utility of MLP and GCN are the same and close to that of LPGNet.

we also train GCNs with the original adjacency matrix, and MLPs without using the adjacency information at all.

7.2.1 Transductive Setting. Figure 5 shows model utility (F1 score) versus privacy budget (ϵ) in transductive setting. We observe that

(1) Compared to DPGCN and LPGNet, Eclipse achieves much better model utility under strong privacy constraints. For instance, given $\epsilon < 1$, both DPGCN and LPGNet suffer drastic performance drops on all datasets (except Facebook). However, Eclipse's performance is not significantly affected.
(2) Under strong privacy constraints with $\epsilon < 1$, Eclipse still achieves better model utility compared to MLP that does not use adjacency information during training. The observations indicate that Eclipse is very effective in preserving graph information under high privacy constraints, leading to overall performance improvement.
(3) When the epsilon value is large ($\epsilon \geq 8$), DPGCN achieves better model utility than Eclipse. However, prior works like LPGNet [21] (Sec 3) demonstrate that the graph in DPGCN is barely perturbed with large epsilons and essentially the same as the original graph. Thus large epsilons offer little privacy protection in the case of DPGCN. In contrast, Eclipse protects privacy by low-rank decomposition and DP noise on singular values. With large epsilons, DP noise fades. However, low-rank decomposition perturbs the graph, and the final model performance is capped at accuracy using a low-rank graph without DP.

Note that for the Facebook dataset, since it is a high homophily graph, nodes with similar features tend to be connected. Therefore, all privacy-preserving training methods give marginal performance improvement by using the edge information from the adjacency matrix. Similar observations also hold on PubMed. Even with the full adjacency matrix, the model utility is marginally improved compared to MLP without using the adjacency matrix. Methods like LPGNet, though, also give slightly better model utility, and significantly compromise protections on edges. On the other hand, Eclipse can bring much-improved privacy protection, with only slightly affecting the model utility, which is more useful in practice.

7.2.2 Inductive Setting. Figure 6 shows model utility versus privacy budget in inductive setting. Eclipse gives a consistent performance under different privacy budgets, which is similar to its performance in transductive setting. However, for LPGNet, due to the highly compressed graph representation, its performance is capped even under moderate privacy constraints, different from that in transductive setting. Surprisingly, the accuracy of DPGCN is stable when privacy budget varies, and even, for Twitch-FR, the accuracy increases when privacy budget becomes smaller, which is in sharp contrast to its performance in transductive setting. This may happen because adding noise to edges may generalize models better for DPGCN. We believe Twitch-FR benefitted from this noise. However, Twitch-DE is an exception where DPGCN's performance quickly collapses with decreasing ϵ. In addition, note that the utility dip at $\epsilon = 6$ is observed on Twitch-ENGB/FR/PTBR in inductive setting except for Twitch-DE dataset. Similar observations are also present in DPGCN [40] (Fig 3(a)). Eclipse targets a setting where edge information is private during training and testing. We also measure accuracy when models are trained using Eclipse but test graphs are unperturbed, and provide the detailed results in Appendix E.

7.3 Resilience Against Privacy Attacks

We further evaluate Eclipse against two link stealing attacks: LPA and LINKTELLER attacks in both transductive and inductive settings. To investigate the model utility versus the attack performance, we adjust the privacy parameter ϵ and respectively evaluate the model's utility and the attacker's performance.

7.3.1 Transductive Setting. For LPA attack, the top row in Figure 7 shows the model utility versus the LPA attack performance in transductive setting. We choose LPA because it infers edges based on the correlation of node features. Note that different edge perturbations can affect LPA result, because LPA uses similarity of posterior rather than raw node features, to analyze potential edges. The posterior is affected by node features, and aggregation operation in GCN model given edge information. Hence, LPA's results depend on the model and perturbation methods on edges. We also perform edge attack using raw node feature similarity, and provide the detailed results in Appendix F. As shown in the top row in Figure 7, the LPA attack gives an AUC higher than 0.5 (i.e. random guess) for MLP which does not use adjacency matrix at all. It indicates that node features and edges share certain common graph information. Therefore, certain connectivity can be inferred via node features. Nevertheless, Eclipse, as well as the baseline methods, aim to protect additional information in the adjacency matrix.

Importantly, Eclipse shows stronger resilience against LPA attacks compared to DPGCN and LPGNet. Specifically, we observe that Eclipse gives a lower AUC under the same F1 score compared to other baselines. On the other hand, if restricting the attack AUC, Eclipse also achieves a higher F1 score. For instance, on the Cora dataset, with an attack AUC of 0.8, Eclipse improves the model utility by 9% compared to DPGCN, and 4% compared to LPGNet.

Note that for the Facebook dataset, due to its high homophily property, the LPA attack recovers most edges by analyzing node features. This indicates that the adjacency matrix of the Facebook graph only encodes common information about the graph, as can be inferred from node features. That is, the adjacency matrix provides little additional *private* information about the graph. For the PubMed dataset, as even the full adjacency matrix provides limited performance gains, we mainly focus on edge protection while slightly trading off model utility.

For LINKTELLER attack, the bottom row in Figure 7 shows the model utility versus the LINKTELLER attack performance. Compared to DPGCN, Eclipse improves resilience against the LINKTELLER attack while maintaining the model utility. On the other hand, LPGNet seems immune to LINKTELLER attack due to no raw graph information used in the model training. However, it does not indicate that LPGNet is effective in protecting the graph information. On the contrary, it is due to the fact that the LINKTELLER attack is limited as opposed to other attacks, such as the LPA attack, that fully explore node features to improve the attack performance. As shown in the top row, with additional knowledge from node features, LPGNet quickly loses its capability of protecting the graph.

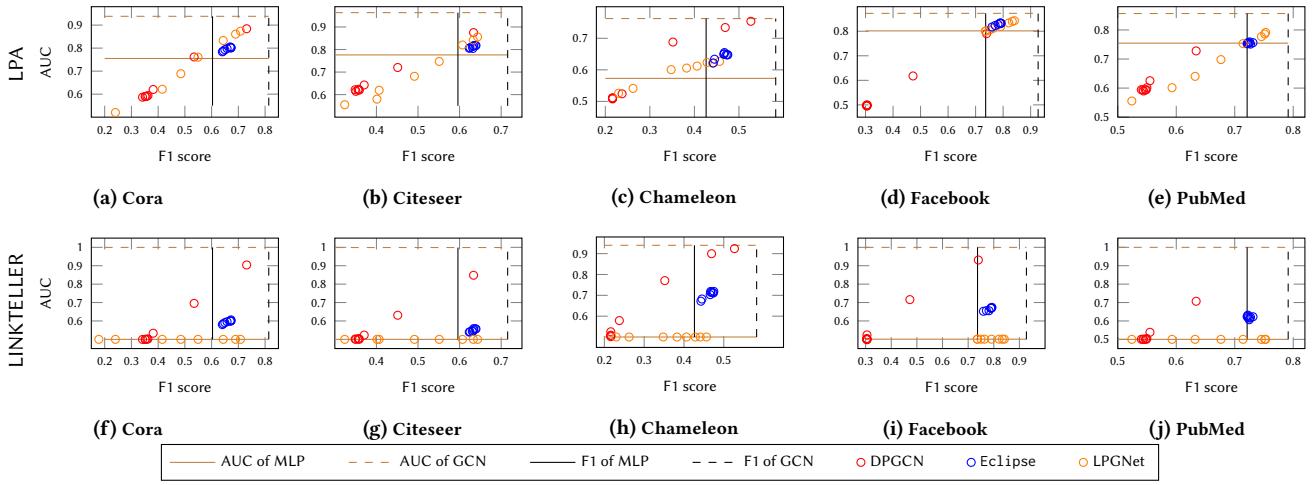

Figure 7: Attack AUC vs. Utility F1 score. in transductive setting. The top row shows LPA attacks, while the bottom row shows LINKTELLER attacks. For LPA attacks, `Eclipse` achieves better model performance with the same attack AUC compared to other baselines. On the other hand, with the same model performance, `Eclipse` shows more resiliency against LPA attacks.

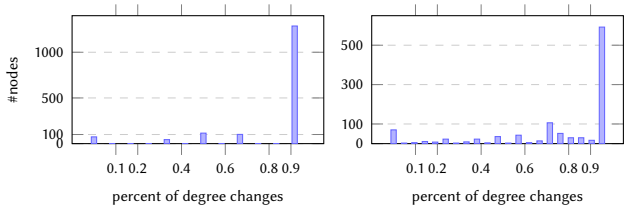

Figure 8: Node degree change on Cora dataset (1621 low degree nodes, 1087 high degree nodes). Close to 80% of low-degree nodes have their node degree modified by 95% or more, while only around 54% of high-degree nodes have their node degree changed by at least 95%.

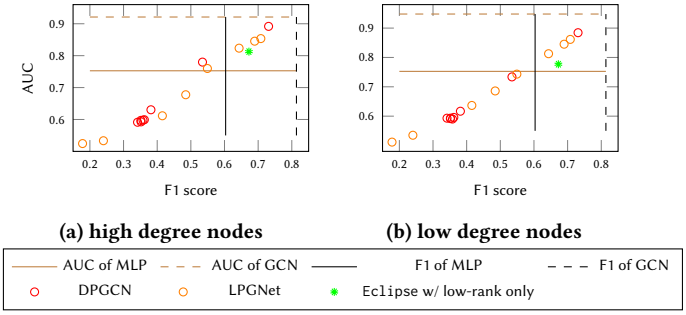

Figure 9: LPA Attack AUC vs. Utility F1 score. in transductive setting breakdown for low and high degree nodes in the Cora dataset.

7.3.2 Attack Performance Breakdown in terms of Low-Degree and High-Degree Nodes. We analyze the effect of low-rank decomposition on nodes with different degrees. Following prior works, low (or high) degree nodes refer to nodes with node degree no larger than (or no smaller than) a threshold value d_{low} (or d_{high}). The values d_{low} and d_{high} are chosen empirically based on the graph. We take Cora dataset as an example and follow the threshold values set for Cora in prior works, where $d_{\text{low}} = 3$ and $d_{\text{high}} = 4$. Based on these threshold values, 1621 out of 2708 nodes in Cora are low-degree nodes, and the remaining 1087 nodes are high-degree nodes. Figure 8 shows the histogram of node degree change for nodes with different node degrees on the Cora dataset. The percentage change in node degree is plotted on the x-axis, and the number of nodes in each bin is shown on the y-axis. With the low-rank decomposition, 80% of low-degree nodes have their node degree reduced by 95% or more, while only around 54% of high-degree nodes are highly affected. Therefore, the low-rank reconstruction already removes substantial number of edges before the DP perturbation. Importantly, it causes more impact on low-degree nodes. Figure 9 shows a detailed LPA attack performance on nodes of different node degrees. We only show LPA attack performance because it is a stronger attack on edges than LINKTELLER attack as discussed in Sec. 7.3.1. First, `Eclipse` with only a low-rank graph (no DP perturbation) already provides edge protection. Importantly, for low- and high-degree nodes, `Eclipse` without DP achieves better model utility than other baselines given the same attack AUC. The reason is that the low-rank decomposition in `Eclipse` preserves the primary topology for better model utility and removes secondary edges in graphs for edge protection. We further observe that LPA attack performs better on high-degree nodes as expected, as a higher percentage of edges from high-degree nodes are preserved in the low-rank adjacency matrix. However, for low-degree nodes, the attack performance is close to the performance of MLP, which does not use the adjacency matrix during training. Therefore, the results show `Eclipse` provides stronger protection on low-degree nodes. For real-world graph data, a high-degree node denotes a centroid in a graph, like a well-known person in social networks. Common DP mechanisms are usually effective in protecting the connectivity of high-degree nodes. However, low-degree nodes are hard to

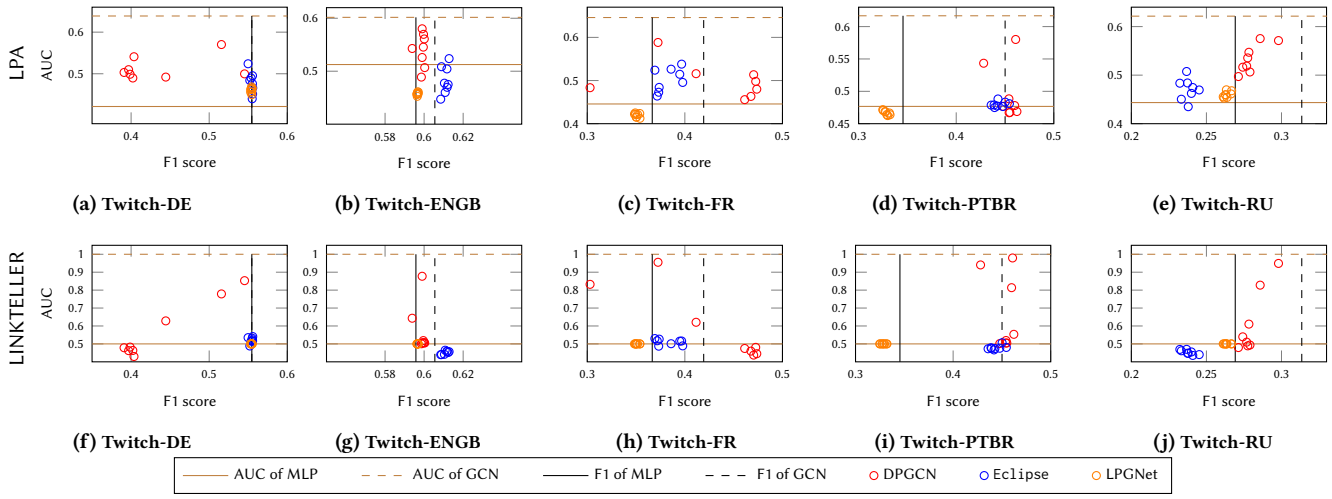

Figure 10: Attack AUC vs. Utility F1 score. in inductive setting for low degree nodes.

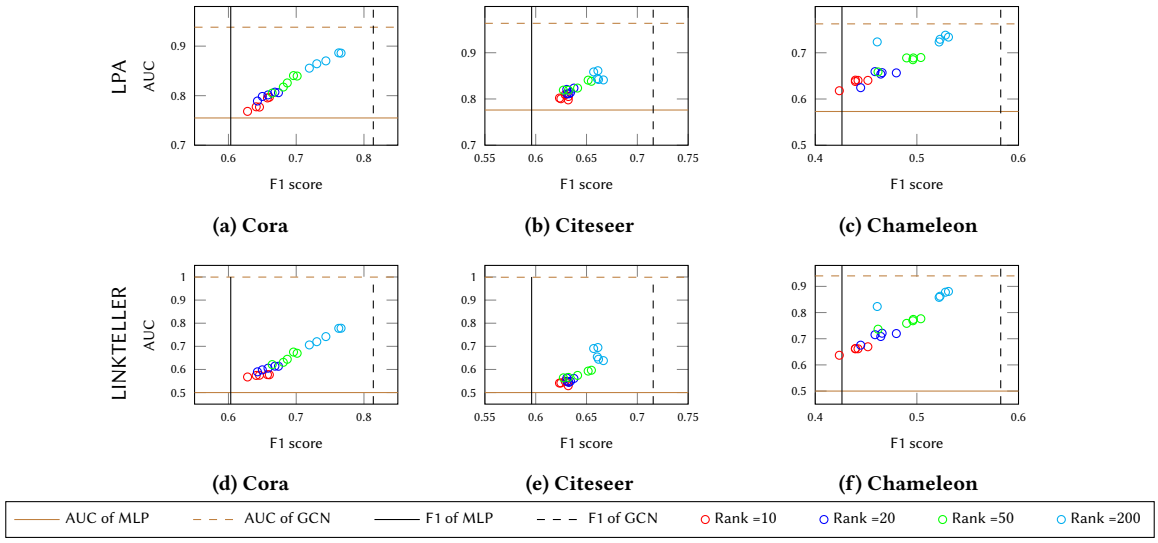

Figure 11: Effect of the rank on Attack AUC vs. F1 score.

protect [30]. Eclipse fills this gap by filtering out connections of low-degree nodes with low-rank reconstruction.

7.3.3 Inductive Setting. Figure 10 shows the attack performance on low-degree nodes in inductive setting. We observe that for most datasets, Eclipse achieves much better model utility while still maintaining similar resilience against LPA and LINTELLER attacks. Specifically, compared to DPGCN, Eclipse archives comparable model utility but with much lower attack AUC, indicating that Eclipse achieves a much better tradeoff between model performance and edge privacy leakage. Unlike LPGNet, which sacrifices significant model performance with low AUC, model performance using Eclipse reaches close to that of GCN with the original adjacency matrix. However, Twitch-RU is an exception where model utility of Eclipse is lower than both DPGCN and LPGNet given similar attack AUC. This may happen because Twitch-RU graph has lower node degree(1-3 edges on average) than other graphs. The low rank clipped many edges causing the graph to lose some topology and thus lower the model utility.

7.4 Effect of Rank

In this experiment, we investigate how changing the rank affects the accuracy/privacy performance of our proposed method Eclipse. We vary rank from 10 to 200 and report attack AUC vs. F1 score under 5 different privacy budgets $\epsilon \in \{0.1, 0.5, 1, 5, 10\}$. We run the target model and the attack model for 5 seeds. The results are shown in Figure 11. We observe that as rank increases, model utility improves. On the other hand, the AUC of LPA and LINKTELLER attack also increases across all three datasets in transductive setting. The reason is that, as we increment rank, more edges are included in

the low-rank graph, leading to potential leakage after DP perturbation. Specifically, during training, more true edges will be involved, resulting in an increasing number of edges in the original graph being reconstructed. As a result, model utility is improved. However, more true edges, that are included in the perturbed adjacency matrix, further reveal the node influence and node feature correlations during training and inference, which compromises the trained model's resilience against attacks. We also observed that while the attack AUC increases with the rank of the perturbed graph, the attack accuracy never exceeds that of a non-private GCN model, meaning that the low-rank reconstruction still removes some of the information in the original graph effectively.

Based on the observation above, we suggest a practical approach for selecting the appropriate rank to train models using `Eclipse` with reasonable attack AUC. For instance, we set an upper threshold (e.g., 0.65) for the attack AUC (e.g., LINKTELLER attack), select the largest possible rank (e.g., 20 for Cora dataset), and train the target model that satisfies the privacy constraint measured by the attack AUC. Then, we evaluate the model utility under the selected rank.

8 CONCLUSION

In this paper, we presented `Eclipse`, a new privacy-preserving GNN training algorithm that ensures edge-level differential privacy for sensitive graph edges. `Eclipse` achieves better privacy-utility tradeoff compared to current state-of-the-art privacy-preserving GNN training methods. `Eclipse` trains GNNs with a low-rank format of the graph via singular value decomposition. The low-rank graph removes many edges from the original graph, while still maintaining enough of the graph structure for better model utility. We further use Gaussian mechanism on the low-rank singular values of the graph's adjacency matrix to protect edges in the low-rank graph. Extensive experiments on real-world graph datasets show that our method achieves good model utility while providing strong privacy protection on edges and outperforms existing methods. We also explored and evaluated the impact of node degree distribution and rank on the attack resilience of our method.

In future work, we will explore providing theoretical justification on the assumption of invariance of principal bases. We will also consider a more adaptive rank selection scheme to further improve privacy-utility tradeoffs when training GNNs. While Eclipse targets scenarios where edge information is private, commonly seen in real-world applications [21, 40], we believe that privacy protection using low-rank decomposition can be extended to node attributes, by exploiting potential low-rank properties in node attributes.

ACKNOWLEDGMENTS

This material is based upon work supported by Defense Advanced Research Projects Agency (DARPA) under Contract Nos. HR001120C0088, NSF award number 2224319, REAL@USC-Meta center, and VMware gift. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *ACM SIGSAC conference on computer and communications security*. ACM, Vienna, 308–318.

[2] Morgane Ayle, Jan Schuchardt, Lukas Gosch, Daniel Zügner, and Stephan Günnemann. 2023. Training Differentially Private Graph Neural Networks with Random Walk Sampling. arXiv:2301.00738 [cs.LG]

[3] Ameya Daigavane, Gagan Madan, Aditya Sinha, Abhradeep Guha Thakurta, Gaurav Aggarwal, and Prateek Jain. 2022. Node-Level Differentially Private Graph Neural Networks. arXiv:2111.15521 [cs.LG]

[4] Ameya Daigavane, Gagan Madan, Aditya Sinha, Abhradeep Guha Thakurta, Gaurav Aggarwal, and Prateek Jain. 2022. Node-Level Differentially Private Graph Neural Networks. arXiv:2111.15521 [cs.LG]

[5] Zulong Diao, Xin Wang, Dafang Zhang, Yingru Liu, Kun Xie, and Shaoyao He. 2019. Dynamic Spatial-Temporal Graph Convolutional Neural Networks for Traffic Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (Jul. 2019), 890–897. https://doi.org/10.1609/aaai.v33i01.3301890

[6] Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. 2021. Quantifying Privacy Leakage in Graph Embedding. In *MobiQuitous 2020 - 17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services* (Darmstadt, Germany) *(MobiQuitous '20)*. Association for Computing Machinery, New York, NY, USA, 76–85. https://doi.org/10.1145/3448891.3448939

[7] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography*, Shai Halevi and Tal Rabin (Eds.). Springer, Berlin, Heidelberg, 265–284.

[8] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[9] Negin Entezari, Saba A. Al-Sayouri, Amirali Darvishzadeh, and Evangelos E. Papalexakis. 2020. All You Need Is Low (Rank): Defending Against Adversarial Attacks on Graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining* (Houston, TX, USA) *(WSDM '20)*. Association for Computing Machinery, New York, NY, USA, 169–177. https://doi.org/10.1145/3336191.3371789

[10] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. Association for Computing Machinery, New York, 417–426.

[11] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, PMLR, Sydney, 1263–1272.

[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017), 1025–1035.

[13] Hanieh Hashemi, Yongqin Wang, and Murali Annavaram. 2021. DarKnight: An Accelerated Framework for Privacy and Integrity Preserving Deep Learning Using Trusted Hardware. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture* (Virtual Event, Greece) *(MICRO '21)*. Association for Computing Machinery, New York, NY, USA, 212–224. https://doi.org/10.1145/3466752.3480112

[14] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. 2021. Stealing links from graph neural networks. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX, Virtual, 2669–2686.

[15] Xinlei He, Rui Wen, Yixin Wu, Michael Backes, Yun Shen, and Yang Zhang. 2021. Node-Level Membership Inference Attacks Against Graph Neural Networks. arXiv:2102.05429 [cs.CR]

[16] Yani Ioannou, Duncan P. Robertson, Jamie Shotton, Roberto Cipolla, and Antonio Criminisi. 2016. Training CNNs with Low-Rank Filters for Efficient Image Classification. In *4th International Conference on Learning Representations, ICLR*, Yoshua Bengio and Yann LeCun (Eds.). OpenReview.net, San Juan, Puerto Rico, 16 pages. http://arxiv.org/abs/1511.06744

[17] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. 2014. Speeding up convolutional neural networks with low rank expansions. *British Machine Vision Conference (BMVC)* 20, 4 (2014), 12 pages.

[18] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. 2014. Private Analysis of Graph Structure. *ACM Trans. Database Syst.* 39, 3, Article 22 (oct 2014), 33 pages. https://doi.org/10.1145/2611523

[19] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2013. Analyzing Graphs with Node Differential Privacy. In *Theory of Cryptography*, Amit Sahai (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 457–476.

[20] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, Toulon, 14 pages. https://openreview.net/forum?id=SJU4ayYgl

[21] Aashish Kolluri, Teodora Baluta, Bryan Hooi, and Prateek Saxena. 2022. LPGNet: Link Private Graph Networks for Node Classification. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (Los Angeles, CA, USA) *(CCS '22)*. Association for Computing Machinery, New York, NY, USA, 1813–1827. https://doi.org/10.1145/3548606.3560705

[22] Krishna Giri Narra, Zhifeng Lin, Yongqin Wang, Keshav Balasubramanian, and Murali Annavaram. 2021. Origami Inference: Private Inference Using Hardware Enclaves. In *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*. 78–84. https://doi.org/10.1109/CLOUD53861.2021.00021

[23] Yue Niu, Ramy E. Ali, and Salman Avestimehr. 2022. 3LegRace: Privacy-Preserving DNN Training over TEEs and GPUs. *Proc. Priv. Enhancing Technol.* 2022, 4 (2022), 183–203. https://doi.org/10.56553/popets-2022-0105

[24] Yue Niu, Ramy E. Ali, Saurav Prakash, and Salman Avestimehr. 2023. All Rivers Run to the Sea: Private Learning with Asymmetric Flows. arXiv:2312.05264 [cs.CR]

[25] Yue Niu, Saurav Prakash, Souvik Kundu, Sunwoo Lee, and Salman Avestimehr. 2023. Federated Learning of Large Models at the Edge via Principal Sub-Model Training. arXiv:2208.13141 [cs.LG]

[26] Iyiola E Olatunji, Mandeep Rathee, Thorben Funke, and M. Khosla. 2023. Private Graph Extraction via Feature Explanations. In *Proceedings on Privacy Enhancing Technologies 2023(2)*. PETs, Lausanne, Switzerland, 59–78. 23rd Privacy Enhancing Technologies Symposium, PETS 2023 ; Conference date: 10-07-2023 Through 15-07-2023.

[27] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. 2020. Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems* 33 (2020), 12559–12571.

[28] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. arXiv:1907.10903 [cs.LG]

[29] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2021. Multi-scale attributed node embedding. *Journal of Complex Networks* 9, 2 (2021), cnab014.

[30] Sina Sajadmanesh, Ali Shahin Shamsabadi, Aurélien Bellet, and Daniel Gatica-Perez. 2023. GAP: Differentially Private Graph Neural Networks with Aggregation Perturbation. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 3223–3240. https://www.usenix.org/conference/usenixsecurity23/presentation/sajadmanesh

[31] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[32] G. W. Stewart. 1990. *Perturbation theory for the singular value decomposition*. Technical Report. University of Maryland, College Park, USA.

[33] Tingting Tang, Ramy E. Ali, Hanieh Hashemi, Tynan Gangwani, Salman Avestimehr, and Murali Annavaram. 2022. Adaptive Verifiable Coded Computing: Towards Fast, Secure and Private Distributed Machine Learning. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 628–638. https://doi.org/10.1109/IPDPS53621.2022.00067

[34] Florian Tramèr and Dan Boneh. 2019. Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware. arXiv preprint arXiv:1806.03287. In *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1806.03287

[35] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. 2020. Traffic Flow Prediction via Spatial Temporal Graph Neural Network. In *Proceedings of The Web Conference 2020* (Taipei, Taiwan) *(WWW '20)*. Association for Computing Machinery, New York, NY, USA, 1082–1092. https://doi.org/10.1145/3366423.3380186

[36] Yongqin Wang, Rachit Rajat, and Murali Annavaram. 2022. MPC-Pipe: an Efficient Pipeline Scheme for Secure Multi-party Machine Learning Inference. arXiv:2209.13643 [cs.CR]

[37] Yongqin Wang, Pratik Sarkar, Nishat Koti, Arpita Patra, and Murali Annavaram. 2023. CompactTag: Minimizing Computation Overheads in Actively-Secure MPC for Deep Neural Networks. arXiv:2311.04406 [cs.CR]

[38] Yongqin Wang, G. Edward Suh, Wenjie Xiong, Benjamin Lefaudeux, Brian Knott, Murali Annavaram, and Hsein-Hsin S. Lee. 2022. Characterization of MPC-based Private Inference for Transformer-based Models. In *2022 IEEE International Symposium on Performance Analysis of Systems and Software*. IEEE Computer Society, Los Alamitos, CA, USA, 187–197. https://doi.org/10.1109/ISPASS55109.2022.00025

[39] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Tao Qi, Yongfeng Huang, and Xing Xie. 2022. A federated graph neural network framework for privacy-preserving personalization. *Nature Communications* 13, 1 (June 2022), 10 pages. https://doi.org/10.1038/s41467-022-30714-9

[40] Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. 2022. LINKTELLER: Recovering Private Edges from Graph Neural Networks via Influence Analysis. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, SAN FRANCISCO, 2005–2024. https://doi.org/10.1109/SP46214.2022.9833806

[41] Hui Xu, Liyao Xiang, Jiahao Yu, Anqi Cao, and Xinbing Wang. 2021. Speedup Robust Graph Structure Learning with Low-Rank Information. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (Virtual Event, Queensland, Australia) *(CIKM '21)*. Association for Computing Machinery, New York, NY, USA, 2241–2250. https://doi.org/10.1145/3459637.3482299

[42] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, New Orleans, 17 pages. https://openreview.net/forum?id=ryGs6iA5Km

[43] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*. PMLR, PMLR, New York, 40–48.

[44] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 974–983. https://doi.org/10.1145/3219819.3219890

[45] Qian Yu, Songze Li, Netanel Raviv, Seyed Mohammadreza Mousavi Kalan, Mahdi Soltanolkotabi, and Salman A Avestimehr. 2019. Lagrange coded computing: Optimal design for resiliency, security, and privacy. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 1215–1225.

[46] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2020. GraphSAINT: Graph Sampling Based Inductive Learning Method. arXiv:1907.04931 [cs.LG]

[47] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 5171–5181.

[48] Zhikun Zhang, Min Chen, Michael Backes, Yun Shen, and Yang Zhang. 2022. Inference attacks against graph neural networks. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX, Boston, 4543–4560.

[49] Zaixi Zhang, Qi Liu, Zhenya Huang, Hao Wang, Chengqiang Lu, Chuanren Liu, and Enhong Chen. 2021. GraphMI: Extracting Private Graph Data from Graph Neural Networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. IJCAI, Virtual, 3749–3755. https://doi.org/10.24963/ijcai.2021/516

[50] Chen Zhe and Aixin Sun. 2021. DP-GCN: Node Classification Based on Both Connectivity and Topology Structure Convolutions for Risky Seller Detection. arXiv:2112.04757 [cs.SI]

[51] Ke Zhou, Hongyuan Zha, and Le Song. 2013. Learning Social Infectivity in Sparse Low-rank Networks Using Multi-dimensional Hawkes Processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, Carlos M. Carvalho and Pradeep Ravikumar (Eds.), Vol. 31. PMLR, Scottsdale, Arizona, 641–649. https://proceedings.mlr.press/v31/zhou13a.html

A PROOF OF THEOREM 1

Theorem 1. *Given A and A' share the same principal bases, with Gaussian noise,*

$$\texttt{Normal}\left(0, \Delta\sqrt{2\ln(1.25/\delta)}/\epsilon_{lr}\right),$$

added to singular values; and Laplacian noise,

$$\texttt{Laplacian}\,(0, 1/\epsilon_e),$$

added to the edge count, the perturbed low-rank A_{plr} using `Eclipse` *satisfies (ϵ, δ)-edge DP with $\epsilon = \epsilon_{lr} + \epsilon_e$.*

Proof. The proof follows the standard Gaussian mechanism and the post-processing and composition rule of the difference privacy [8]. `Eclipse` consists of two randomized mechanisms: A Gaussian mechanism on singular values and a Laplacian mechanism on the edge count. First, we show that perturbed singular values $\tilde{s}$ is (ϵ_{lr}, δ)-edge differentially private.

Given sensitivity Δ, with Gaussian noise $\texttt{Normal}(0, \frac{\Delta\sqrt{2\ln(1.25/\delta)}}{\epsilon_{lr}})$ adding to singular values, $\tilde{s}$ (line 9 in Algorithm 1) is (ϵ_{lr}, δ)-edge differentially private based on the standard Gaussian mechanism. When obtaining a low-rank A_{lr} (line 11 in Algorithm 1), as it is the post-processing given $\tilde{s}$ and shared principal bases, it does not change the differential privacy budget. Hence, the low-rank reconstruction follows the same privacy budget as $\tilde{s}$.

Then, for the Laplacian mechanism on the edge count (line 5 in Algorithm 1), $\tilde{E}$ satisfied $(\epsilon_e, 0)$-edge differentially private.

Given a low-rank A_{lr} and the edge count $\tilde{E}$, we apply binary quantization (line 16 in Algorithm 1) to obtain the final low-rank adjacency matrix A_{plr}. By the composition theorem and the post-processing rule of differential privacy, we obtain the final privacy guarantee

$$\epsilon = \epsilon_{lr} + \epsilon_e, \quad \delta = \delta + 0.$$

Hence, the perturbed low-rank A_{plr} using Eclipse guarantees (ϵ, δ)-edge differential privacy. □

B COSINE SIMILARITY OF PRINCIPAL BASES FOR OTHER DATASETS

Figure 12 show the cosine similarity of principal bases between adjacent graphs on the larger Facebook dataset. Similar as Figure 2, principal bases between adjacent graphs are almost the same.

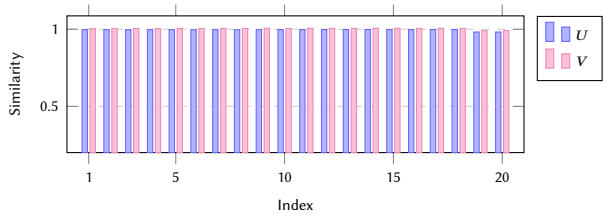

Figure 12: Cosine similarity of principal basis vectors in the Facebook dataset.

C PRIVACY-UTILITY TRADEOFF COMPARED WITH PRIVGRAPH

PrivGraph is a graph synthesis algorithm that exploits community information to publish a synthetic graph under DP guarantee. PrivGraph aims to reduce the perturbation noise to the adjacency matrix while limiting the information loss during encoding the graph data, which is similar to the goal of Eclipse. To compare Eclipse's perturbation method to PrivGraph, we follow PrivGraph's procedure to generate a synthetic graph and use it to train a GCN model on Chameleon (the common dataset used in Eclipse and PrivGraph). We try 4 different privacy budgets $\epsilon \in \{0.5, 1, 2, 3\}$ following PrivGraph's experimental setup, and report the GCN model's test accuracy in Figure 13. We found that PrivGraph outperforms DPGCN which perturbs the adjacency matrix directly. However, given the same privacy budget, PrivGraph achieves lower test accuracy than Eclipse. One reason is that PrivGraph relies on *community information* when reconstructing edges. However, Chameleon is considered a heterophilous dataset where the features and edges do not correlate well with the ground-truth labels (clusters/communities). The community detection algorithm used in PrivGraph cannot effectively capture connection information for such heterophilous graph. On the other hand, Eclipse uses the low-rank method to preserve the primary topology, which is independent of how closely edges and community information correlate.

D WHEN WILL THE ASSUMPTION FAIL?

Consider a fully connected graph G_F consisting of 3 nodes only, with its adjacency matrix denoted as A_F. Upon SVD, we obtain the singular vectors and singular values as shown below. To obtain its neighboring graph G'_F, we remove the edge between node 1 and node 2, the resulting adjacency matrix A'_F and its singular vectors and singular values can be found as shown below. We can see that G_F and G'_F do not share principal bases. However, such extreme cases, where adding/deleting one edge results in completely different principal bases, are not observed in real-world datasets used in this paper.

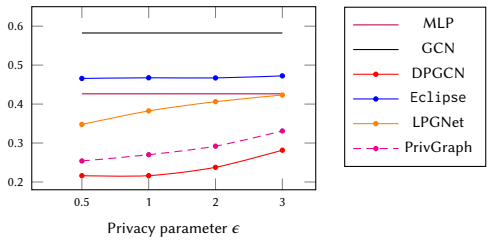

Figure 13: Model utility on Chameleon dataset. Eclipse achieves higher accuracy compared to DPGCN, LPGNet, and PrivGraph, while PrivGraph performs better than DPGCN.

$$A_F \xrightarrow{SVD} U_F \cdot S_F \cdot V_F$$

$$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \xrightarrow{SVD}$$

$$\begin{bmatrix} 0.58 & -0.71 & -0.41 \\ 0.58 & 0.71 & -0.41 \\ 0.58 & 0 & 0.82 \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.58 & 0.71 & -0.41 \\ 0.58 & -0.71 & -0.41 \\ 0.58 & 0 & -0.82 \end{bmatrix}$$

$$A'_F \xrightarrow{SVD} U'_F \cdot S'_F \cdot V'_F$$

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \xrightarrow{SVD}$$

$$\begin{bmatrix} 0.71 & 0 & -0.71 \\ 0.71 & 0 & 0.71 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1.41 & 0 & 0 \\ 0 & 1.41 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0.71 & -0.71 \\ 0 & 0.71 & 0.71 \\ 1 & 0 & 0 \end{bmatrix}$$

E UNPERTURBED TEST GRAPH

Eclipse targets a setting where edge information is private during training and testing, commonly seen in edge-DP literature [21, 40]. We also measure accuracy when models are trained using Eclipse but test graphs are unperturbed and report the test accuracy in Table 2. We use Twitch dataset for inductive settings where training and test graphs differ. We observe that Eclipse is generally robust to test graph perturbations. Both unperturbed and perturbed graphs provide similar performance.

F ATTACK USING NODE FEATURE SIMILARITY

We perform edge attack using raw node feature similarity and report the attack AUC as well as the AUC drop compared to LPA in Table 3. Not surprisingly using only node features, attack AUC is

generally lower than that achieved with LPA (brown-dashed lines Figure 7), using model output similarity.

Table 2: Test Accuracy when test graphs are unperturbed and perturbed.

Dataset	ϵ	Acc. w/o perturbation	Acc. w/ perturbation
TwitchDE	0.1	0.567	0.55
	0.2	0.571	0.552
	0.5	0.569	0.555
	1	0.567	0.554
	2	0.567	0.556
	4	0.567	0.556
	6	0.567	0.556
	8	0.568	0.555
TwitchFR	0.1	0.379	0.395
	0.2	0.378	0.397
	0.5	0.375	0.398
	1	0.374	0.386
	2	0.35	0.369
	4	0.354	0.372
	6	0.349	0.373
	8	0.348	0.374
TwitchRU	0.1	0.226	0.245
	0.2	0.224	0.24
	0.5	0.229	0.232
	1	0.223	0.238
	2	0.221	0.233
	4	0.223	0.237
	6	0.224	0.237
	8	0.227	0.241
TwitchENGB	0.1	0.618	0.609
	0.2	0.618	0.609
	0.5	0.619	0.61
	1	0.619	0.611
	2	0.619	0.612
	4	0.619	0.612
	6	0.619	0.612
	8	0.619	0.613
TwitchPTBR	0.1	0.395	0.45
	0.2	0.396	0.455
	0.5	0.381	0.448
	1	0.363	0.443
	2	0.369	0.439
	4	0.372	0.436
	6	0.374	0.442
	8	0.387	0.439

Table 3: Attack AUC using node feature similarity.

Dataset	AUC	AUC drop v.s. LPA
Cora	0.821	-0.12
Citeseer	0.902	-0.06
Chameleon	0.571	-0.19
PubMed	0.902	0.04
Facebook	0.636	-0.24