GCL-Leak: Link Membership Inference Attacks against Graph Contrastive Learning

Xiuling Wang
Stevens Institute of Technology
Hoboken, NJ, USA
xwang193@stevens.edu

Wendy Hui Wang
Stevens Institute of Technology
Hoboken, NJ, USA
hwang4@stevens.edu

ABSTRACT

Graph contrastive learning (GCL) has emerged as a successful method for self-supervised graph learning. It involves generating augmented views of a graph by augmenting its edges and aims to learn node embeddings that are invariant to graph augmentation. Despite its effectiveness, the potential privacy risks associated with GCL models have not been thoroughly explored. In this paper, we delve into the privacy vulnerability of GCL models through the lens of link membership inference attacks (LMIA). Specifically, we focus on the federated setting where the adversary has white-box access to the node embeddings of all the augmented views generated by the target GCL model. Designing such white-box LMIAs against GCL models presents a significant and unique challenge due to potential variations in link memberships among node pairs in the target graph and its augmented views. This variability renders members indistinguishable from non-members when relying solely on the similarity of their node embeddings in the augmented views. To address this challenge, our in-depth analysis reveals that the key distinguishing factor lies in the similarity of node embeddings within augmented views where the node pairs share identical link memberships as those in the training graph. However, this poses a second challenge, as information about whether a node pair has identical link membership in both the training graph and augmented views is only available during the attack training phase. This demands the attack classifier to handle the additional "identical-membership" information which is available only for training and not for testing. To overcome this challenge, we propose GCL-Leak, the first link membership inference attack against GCL models. The key component of GCL-Leak is a new attack classifier model designed under the "Learning Using Privileged Information (LUPI)" paradigm, where the privileged information of "same-membership" is encoded as part of the attack classifier's structure. Our extensive set of experiments on four representative GCL models showcases the effectiveness of GCL-Leak. Additionally, we develop two defense mechanisms that introduce perturbation to the node embeddings. Our empirical evaluation demonstrates that both defense mechanisms significantly reduce attack accuracy while preserving the accuracy of GCL models.

Proceedings on Privacy Enhancing Technologies 2024(3), 165–185

https://doi.org/10.56553/popets-2024-0073

KEYWORDS

Membership inference attack; Graph constrastive learning; Self-supervised learning; Privacy attacks and defense; Trustworthy machine learning.

1 INTRODUCTION

Graph learning has gained significant popularity in various fields. However, training graph learning models with limited labeled data poses a significant challenge, especially in domains like biology and chemistry, where task-specific labels are scarce [21, 87]. To address this issue, self-supervised learning (SSL) graph models [24, 30, 64, 66] have emerged as a promising approach.

Among various types of SSL graph models, *graph contrastive learning* (GCL) models have become prevalent [58, 64, 84]. At a high level, given a training graph G, a GCL model generates multiple augmented views of G through graph augmentation, aiming to learn the graph representation (node embedding) of G that is invariant to augmentation. Typical augmentation functions modify the graph topology by adding or deleting edges in the augmented views [14, 85, 86]. Recently, GCL has been extended to the federated setting [4, 59, 68] where the clients share the node embeddings of the augmented views obtained from their local graphs with the server, while the server aggregates these local embeddings to learn the global embedding.

Threat model. While substantial efforts have been dedicated to enhancing the performance of GCL models [14, 71, 75, 85], the privacy vulnerabilities associated with these models have been largely overlooked. This paper primarily focuses on *Link Membership Inference Attacks* (LMIAs) against GCL models which aim to infer whether a specific node pair is connected in the training graph of the target model. We consider an adversary situated in the federated setting, potentially an honest-but-curious server [55, 68]. The adversary leverages node embeddings within all augmented views obtained through white-box access to the target model for LMIA inference. Given that graph links can represent sensitive information such as private social relationships and associations between diseases and specific patients, the disclosure of these links raises significant privacy concerns.

Analysis: Several existing methods [8, 17, 63, 80] have investigated the privacy vulnerabilities of supervised graph models, particularly Graph Neural Networks (GNNs), against LMIAs. These investigations commonly rely on the similarity of node embeddings to discern between linked node pairs (members) and non-linked ones (non-members). However, through an extensive analysis of four mainstream GCL models and three real-world graphs, we revealed that these existing methods cannot be directly applied or easily adapted to GCL models. This is because the differentiation

between member and non-member links cannot be solely based on the similarity of their embeddings in augmented views (refer to Section 4). Instead, the attack model should account for the edge membership in both the training graph and the augmented views. Our analysis specifically indicates that the distinguishability between members and non-members relies exclusively on the embedding similarity within augmented views featuring "identical-membership"—meaning the node pairs in these augmented views share identical link memberships as those in the training graph.

Attack: Building upon the insights obtained from our analysis, we introduce GCL-Leak, the first LMIA tailored for GCL models. GCL-Leak leverages a "shadow graph", acquired from external sources, providing access to augmented views and the corresponding node embeddings within these views through white-box access to the target GCL model. In the subsequent steps, it generates attack training data, extracting features from the embedding similarity of node pairs within augmented views that have the "identical-membership" with the shadow graph. However, this raises a new challenge in designing the attack classifier. During the inference phase, the adversary cannot extract the "identical-membership" information for the target node pairs without access to the original training graph and its associated augmented views. To address this challenge, we design a new attack classifier model under the *Learning Using Privileged Information (LUPI)* paradigm [42, 56] which enables learning from the privileged information available during training but not during testing. Specifically, the attack features take the contrastive nature of GCL models into consideration, with the "identical-membership" information encoded as part of the structure of the attack classifier.

Evaluation: We evaluate the effectiveness of GCL-Leak through extensive empirical evaluations. The results demonstrate that GCL-Leak outperforms the existing LMIAs across various settings, achieving an attack accuracy in the range of 0.83 to 1. Moreover, GCL-Leak remains effective in the transfer setting where the shadow and target graphs are sampled from different domains and distributions. Moreover, our empirical assessment reveals that GCL-Leak surpasses not only attacks lacking LUPI but also those incorporating LUPI through alternative methodologies [6, 61].

Defense: We propose two defense mechanisms named FNoise and PNoise to mitigate the vulnerabilities of GCL models against GCL-Leak. Both defense methods introduce Laplace noise to node embeddings. However, FNoise incorporates full noise across all dimensions of the embedding, whereas PNoise selectively adds noise to dimensions considered of lesser importance to the model. Through evaluation, we demonstrate that both defense methods significantly reduce the attack accuracy while preserving the accuracy of GCL models. In addition, both methods outperform the existing defense solutions in terms of the trade-off between defense power and model accuracy.

Our contributions. Our main contributions are as follows [1]

- We conduct empirical studies and analysis to uncover the reasons behind the limitations of existing LMIAs when targeting GCL models. We also gain new insights that can be utilized to improve the effectiveness of LMIAs against GCL models.

[1]Our code and datasets are available at the link: https://gitfront.io/r/user-8658281/66TPGy4ACgzd/MIA-GCL/

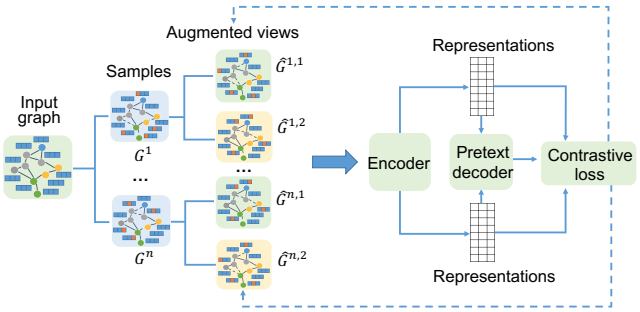


Figure 1: An overview of GCL models.

- We design GCL-Leak, the first LMIA against GCL models. Our empirical evaluation demonstrates that GCL-Leak is highly accurate in link inference against GCL models.
- We devise two defense mechanisms against GCL-Leak and conduct a thorough empirical evaluation to demonstrate their effectiveness.

2 PRELIMINARIES

2.1 Graph Contrastive Learning

Given a graph $G(V, E)$, where V and E denote the vertices and edges respectively, a Graph Contrastive Learning (GCL) model aims to learn graph representations of G that are invariant to a set of manually specified transformations known as *augmentations*. A GCL framework typically consists of the following key components: (1) the *graph augmentation function* that generates multiple views from the given graph; (2) an *encoder* f_θ (parameterized by θ) that is responsible for learning low-dimensional representations (embeddings) of the graph; (3) a *pretext decoder* p_ϕ (parameterized by ϕ) that estimates the agreement between any two augmentations; and (4) the *contrastive objective* that encourages consistency among positive pairs and inconsistency among negative pairs. Figure 1 illustrates the GCL framework. Next, we briefly explain the details of each component.

Graph augmentation functions. Given a graph sample G, the GCL model generates a number of *augmented views* of G by applying the graph augmentation function on G. The existing graph augmentation functions can be categorized into two types: *topology augmentation* and *feature augmentation* [7, 81].

- **Topology augmentation**: Topology augmentation functions encompass three categories: (1) *node augmentation* involves removing a set of nodes and their associated edges or inserting fake nodes into the graph; (2) *edge augmentation* entails adding new edges that do not exist in the original graph and/or removing a fraction of edges from the original graph; and (3) *subgraph augmentation* is a hybrid approach that combines node and edge augmentation techniques.
- **Feature augmentation**: there are two types of feature augmentation functions: (1) *node feature masking* that randomly masks a fraction of features in the graph by setting them to a neutral value; (2) *node feature shuffle* involves swapping the features of some nodes in the graph.

In this paper, we mainly focus on edge augmentation as it has been used by most of the GCL models. While only a few existing

GCL models incorporate both edge removal and addition, we consider either edge removal or edge addition, but not both, as the edge augmentation method. The degree of edge augmentation is determined by an *augmentation ratio*, where a higher ratio indicates a greater number of edges to be inserted or deleted.

The encoder-decoder framework with the contrastive objective. To capture crucial global graph information, GCL aims to learn graph representations that are invariant to small perturbations. This is achieved by maximizing the agreement between augmented views of the same input graph through a contrastive loss in the latent space. Given a training graph G, a minibatch of n graphs $\{G^1, \ldots, G^n\}$ is randomly sampled from G and processed using contrastive learning. In this paper, we follow the literature and consider two augmented views (denoted as $\hat{G}^{i,1}$ and $\hat{G}^{i,2}$) for each input graph sample G^i.

For any input graph sample G^i and any node $v \in G^i$, its embedding generated in the augmented view $\hat{G}^{i,1}$ is treated as the anchor. The embedding of $v \in \hat{G}^{i,2}$ forms the *positive sample* of v, while the embedding of any node v' ($v' \neq v$) in the two views is regarded as a *negative sample* of v. GCL can be formalized as an encoder-decoder framework in which the encoder f_θ can be any GNN model [13, 25, 57] that learns a low-dimensional node representation (embedding) for each input augmented view. The decoder p_ϕ distinguishes the embeddings of positive and negative pairs. The contrastive objective of the GCL models is to maximize the agreement between positive pairs and minimize the agreement between negative pairs, formalized as follows:

$$\theta^*, \phi^* = \min_{\theta,\phi} \mathcal{L}(p_\phi(f_\theta(\hat{G}^{i,1}), f_\theta(\hat{G}^{i,2}))), \tag{1}$$

A commonly used loss function $\mathcal{L}$ is the mutual-information based loss (Jensen-Shannon divergence [52]).

In this paper, we investigate four representative contrastive-based GCL models, namely **GRACE** [85], **CGA-SSG** [75], **GCA** [86] and **MVGRL** [14]. Table 1 provides an overview of the major differences between these models. These models employ different contrastive objectives. In particular, GRACE, MVGRL, and GCA focus on the node-level contrastive loss, while CCA-SSG considers the contrastive objectives at both node and feature levels. In Section 6, we will discuss how different contrastive objectives impact the attack performance.

Impact of edge augmentation on edge membership. The four representative GCL models (GRACE, CGA-SSG, GCA, MVGRL) employ different edge augmentation functions. Specifically, GRACE & CCA-SSG randomly remove edges, assuming equal importance of all edges. GCA removes edges based on their weights, with lower-weight edges having a higher probability of removal. And MVGRL performs *edge addition* by connecting *indirectly* connected node pairs. The selection of new edges is based on their weights, with higher-weighted node pairs having a higher probability of connection. The weight of each node pair is computed using personalized PageRank [41].

Different augmentation functions cause distinct changes in edge membership within augmented views: edge removal augmentation converts member edges in the original training graph to non-members in the augmented views, while edge addition augmentation changes non-member edges in the training graph to members in the augmented views. We will analyze the impact of different augmentation functions on the attack performance in Section 6.

2.2 GCL under Federated Setting

Recently, GCL models have been extended to the federated learning (FL) setting [4, 53, 59, 68]. In this framework, multiple clients and a central server are involved. Each client possesses a private graph that cannot be shared with other clients or the server due to privacy concerns. The server holds a global GCL model and distributes it to all clients as the initialization for their local GCL models. Each client trains its local GCL model using its private graph and obtains the embeddings of all the augmented views. Then each client sends these local embeddings to the server. The server aggregates the clients' local embeddings to generate the global embedding by utilizing an aggregation mechanism such as FedAvg [34], self-adaptive method [59], and clustering [65]. Next, the server sends the updated global embeddings back to the clients. This learning process iterates until the global model reaches convergence.

2.3 Learning Using Privileged Information

Learning Using Privileged Information (LUPI) is a novel learning paradigm that extends traditional training examples by incorporating additional privileged information. This privileged information, which may include qualities, properties, or contextual details of the training instances, is available during the training phase but not during testing. In the LUPI paradigm, along with the standard training data, the learner is provided with privileged information x^*. The learning goal is to utilize the privileged information x^* to learn a better classifier than one would learn without it.

Several approaches have been proposed for classifier learning under the LUPI paradigm [3, 56, 61, 62]. These approaches represent privileged information in different ways: (1) the privileged information is represented as a set of slack functions. The learner simultaneously estimate the decision function h and the slack functions [56]; (2) the privileged information is combined with the classifier as regularization terms. The model is penalized if its loss, considering the privileged information, exceeds the loss when the privileged information is not considered [62, 69]; and (3) the privileged information is represented as secondary targets. The classifier model first classifies the secondary targets and then uses the predictions to classify with primary features [61]. However, these approaches are closely tied to convex optimization problems. We will discuss how to handle the privileged information in our setting (Section 5).

3 PROBLEM FORMULATION

3.1 Threat Model

Given a GCL model Φ, its training graph $G(V, E)$, where V and E denote the vertices and edges respectively, for any node pair v_j and v_k ($v_j, v_k \in V$), the adversary aims to infer if there is an edge (v_j, v_k) in G. In this paper, we consider the server as the adversary, and assume he has the following adversary knowledge:

- **White-box access to target model** Φ: Under the federated setting, the adversary has the white-box access to Φ, including the architecture of both encoders and decoders, parameters, and the contrastive loss function of Φ. The adversary also can access

GCL Model	Edge Augmentation	Contrastive objective	Change of edge membership in aug. views
GRACE [85]	Edge removal (uniformly)	Node-level	Member (Input.G) → non-member (view)
CCA-SSG [75]		Node & feature-level	
GCA [86]	Edge removal (non-uniform)	Node-level	Member (Input.G) → non-member (view)
MVGRL [14]	Edge addition (non-uniform)	Node-level	Non-member (Input.G) → member (view)

Table 1: Comparison of the four GCL models used in the paper.

the local node embeddings of all the augmented views generated by each client.

- **Shadow graph** G^S: the adversary has access to a shadow graph (or multiple graphs) G^S which contains its own structure and node attributes. G^S may be sampled from a different domain and distribution from those of G. The shadow graphs can be obtained from the data repositories that are publicly available. [2]

In addition to the shadow graph, the adversary may also possess knowledge of a subgraph within the target graph G, which can be considered as a special case of the shadow graph.

An example of a white-box attack is the federated GCL paradigm, where clients share the node embeddings of their local augmented views with a server (acting as the adversary). The server aims to infer the structural information present in the clients' private graphs [4, 53, 55, 68]. Even in situations where the requirements for a white-box attack may not be practical for an adversary, the ability to launch a more powerful attack could be valuable for designing more robust defenses [26].

Formally, given a training graph G, a GCL model Φ trained on G, a target node pair v_j, v_k ($v_j, v_k \in G$) and their embeddings $\{z_j\}, \{z_k\}$ in all the augmented views of G, as well as the adversary's auxiliary knowledge K_{Aug}, the link membership inference attack (LMIA) can be defined as a mapping function f:

$$f : v_j, v_k, \{z_j\}, \{z_k\}, K_{Aug} \rightarrow \{0, 1\} \quad (2)$$

where 1 (0, resp.) indicates the edge $(v_j, v_k) \in G$ ($(v_j, v_k) \notin G$, resp).

4 PRE-ATTACK ANALYSIS: DISTINGUISHABILITY OF MEMBERSHIP

There are notable distinctions between the GCL setting and the supervised setting in terms of edge membership. In supervised learning, all node pairs have fixed membership during training. Conversely, for GCL models, the link membership of node pairs can vary across different augmented views and may differ from the original training graph. This introduces two types of link membership: (1) the **ground-truth membership** which represents the link membership of a node pair in the original training graph; and (2) the **augmented membership** which denotes the link membership of a node pair in a particular augmented view. Each node pair possesses a fixed ground-truth membership, but it can have different augmented memberships across different augmented views due to edge augmentation.

There is a relationship exists between augmented and ground-truth membership. Since most GCL models only introduce minimal perturbations to the graph topology, the majority of augmented memberships across all augmented views remain consistent with the ground-truth membership for each node pair. Consequently, the ground-truth membership can be inferred from the augmented memberships using majority voting: the augmented membership type that appears most frequently across all augmented views is considered the ground-truth membership.

The existing LMIA [8] that leverages node embeddings for inference has identified an important property of edge membership for the supervised graph learning models: member and non-member links can be distinguished by the similarity of node embeddings as the connected nodes (members) tend to have more similar embeddings compared to disconnected nodes (non-members) on average. However, as the adversary only has access to the embeddings of the augmented views, two crucial questions arise: (1) *Are ground-truth member/non-member links distinguishable by the embedding similarity of node pairs in the augmented views?* (2) *If (1) does not hold, as the ground-truth membership can be inferred from the augmented membership, are augmented member/non-member links distinguishable by the embedding similarity of node pairs in augmented views?*

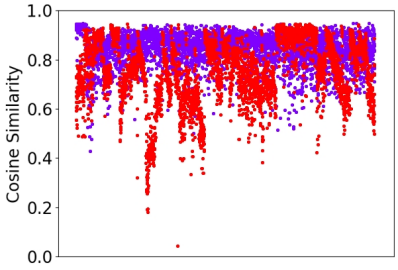
To answer these two questions, we trained the four representative GCL models (GRACE, CGA-SSG, GCA, MVGRL) on three real-world datasets [3] and measured the Cosine similarity between the embeddings of the node pairs in the augmented views. Then we assign either the *ground-truth membership* label or the *augmented membership* label to each node pair and visualize the distribution of embedding similarity for each membership type. Figure 2 visualizes the distribution of members/non-members when GRACE is the target model. The results of other GCL models can be found in Appendix A. We have the following four observations.

Obs.1 – Augmented member/non-member edges are not distinguishable by embedding similarity in the augmented views. Figure 2 (a) visualizes the embedding similarity of the augmented member and non-member edges for the Cora dataset. It is evident that members and non-members in the augmented views cannot be distinguished based on their embedding similarity. This observation can be attributed to the nature of GCL models, which aim to learn graph representations that are invariant to small topology perturbations. As a result, the node embeddings and their similarity are expected to remain unchanged despite membership variations in the augmented views.

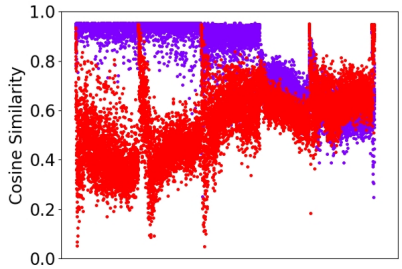
Obs.2 – Ground-truth member/non-member edges are partially distinguishable by embedding similarity in the augmented views. Figure 2 (b) visualizes the embedding similarity of the node pairs in all the augmented views where each node pair is associated with its ground-truth membership label. While ground-truth members/non-members exhibit better distinguishability than augmented members/non-members (as observed in Observation 1), there are still overlaps in certain regions. In other words, the similarity between some member node pairs is close to or smaller than that of some non-member node pairs. This observation aligns with

[2] Examples of public graph data repositories include Stanford's SNAP dataset (http://snap.stanford.edu/data/index.html) and UCI graph repository (http://networkdata.ics.uci.edu/index.html).

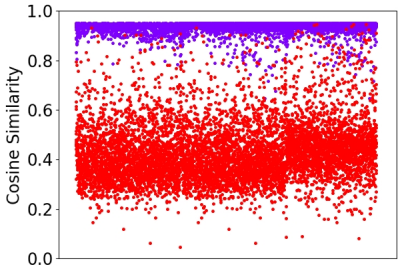
[3] The three datasets are Cora, Citeseer, and Facebook datasets. More details of the datasets can be found in Section 6.1.



(a) Distribution of augmented membership labels by similarity of node embeddings in augmented views.



(b) Distribution of ground-truth membership labels by similarity of node embeddings in augmented views.



(c) Dist. of ground-truth membership labels by similarity of embeddings in augmented views of "identical membership".

Figure 2: Distribution of augmented/ground-truth members/non-members by embedding similarity (Cora dataset). Members and non-members are denoted in blue and red respectively. X-axis: member and non-member samples, y-axis: Cosine similarity between node embeddings.

our intuition. Although graph representations should be invariant to small topology perturbations, the embedding similarity in the augmented views still contains noise due to edge augmentation. Thus, in an augmented view, a member node pair (v_j, v_k) can be similar to a non-member node pair (v'_j, v'_k) in terms of their embedding similarity, due to the removal of the edge (v_j, v_k) or insertion of a new edge (v'_j, v'_k).

Obs.3 – Ground-truth members/non-members are well distinguishable by the embedding similarity of those whose augmented membership is the same as their ground-truth membership. Figure 2 (c) visualizes the embedding similarity of the node pairs in the augmented views where their augmented membership is the same as their ground-truth membership. Each node pair is associated with its ground-truth membership. We observe that, by removing those similarity values of node pairs of inconsistent augmented and ground-truth membership, the ground-truth members/non-members are well distinguishable by the embedding similarity in the augmented views. We will utilize this observation to design the attack features.

Obs.4 – Obs.1 - 3 hold for both edge removal and edge addition augmentations. While Figure 2 (a) - (c) shows the embedding of the GRACE model (edge removal as augmentation), we observe the same phenomenon on the embedding of MVGRL model (edge addition as augmentation) (Figure 13 in Appendix A).

5 DETAILS OF GCL-LEAK

Our pre-attack analysis (Section 4) reveals important observations. First, Obs.1 suggests using the ground-truth membership as the label for training the attack model instead of the augmented membership. Second, Obs.2 & 3 indicate that only node pairs with consistent augmented and ground-truth memberships should be selected for feature extraction. Based on these insights, we design GCL-Leak, the first LMIA against GCL models. In this section, we present the details of GCL-Leak.

Attack overview and challenges. At a high level, GCL-Leak employs a shadow graph, denoted as G^S, to train the membership inference attack model. The adversary leverages its white-box access to the target model to input G^S and retrieve the embeddings of all node pairs in the augmented views of G^S, along with their augmented and ground-truth memberships. Subsequently, the adversary conducts supervised training of the attack model using node pairs and their membership labels extracted from G^S. Finally, the trained attack model is utilized to infer the membership of nodes as members or non-members in the target graph, denoted as G.

A naive approach is to extract attack features from the embedding similarity of those node pairs which have consistent augmented and ground-truth memberships. While this is a valid approach for generating the attack training data (denoted as A^{train}), it poses a challenge on generating the attack testing data (denoted as A^{test}), as the adversary lacks the "same-membership" information for the target node pairs. This absence of "same-membership" information information during testing prevents its combination with the input features to predict the membership label.

To address this challenge, we design our attack classifier under the *Learning Using Privileged Information (LUPI)* paradigm [42, 56]. At a high level, LUPI leverages the privileged information that is available during training but not during testing. In our context, privileged information refers to the "same-membership" information. Specifically, we encode this "same-membership" information into the attack classifier features to construct an effective attack classifier.

Essentially, GCL-Leak includes three phases: (1) *generating MIA training data*, (2) *attack model training*, and (3) *membership inference*. Figure 3 illustrates the framework of GCL-Leak. In the following subsections, we delve into the details of each phase of GCL-Leak.

5.1 Notations and Definitions

Before we delve into the details of the attack, we define some concepts and notations that will be commonly used in the paper. Given a graph sample G^i and an augmented view $\hat{G}^i$ of G^i, we use z^i_v to denote the embedding of a node $v \in \hat{G}^i$. Given a graph sample G^i, two nodes $v_j, v_k \in G^i$, and an augmented view $\hat{G}^i$ of G_i, we say the node pair (v_j, v_k) has consistent membership in $\hat{G}^i$ if the augmented membership of (v_j, v_k) in $\hat{G}^i$ is identical to the ground-truth membership of (v_j, v_k) in G^i. We say the node pair (v_j, v_k) is a *true positive member* in $\hat{G}^i$ if v_j and v_k are connected in both $\hat{G}^i$ and G^i, and (v_j, v_k) is a *true negative member* in $\hat{G}^i$ if v_j and v_k are connected in either $\hat{G}^i$ or G^i, but not both. We say a node pair (v_j, v_k) is a *consistent true positive* member if v_j and v_k are

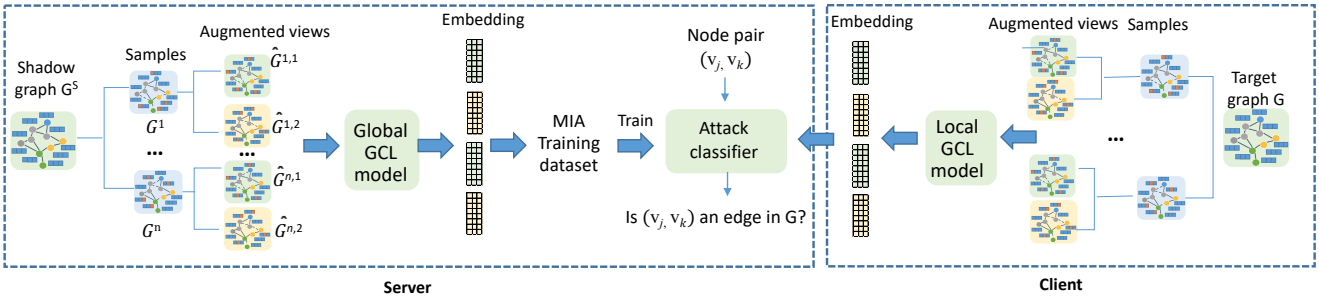


Figure 3: An overview of GCL-Leak.

connected (member) in not only G^i but also all the augmented views of G^i, and a *consistent true negative* member if they are disconnected (non-member) in neither G^i nor any augmented view of G^i.

5.2 Phase 1: Generating Attack Training Data

We present two types of methods to construct the attack training datasets: (1) The *basic method* that constructs the attack training data from the node pairs in all augmented views; and (2) the *enhanced method* that constructs the attack training data from the node pairs in the augmented views that respect an additional contrastive constraint of the target GCL model. For simplicity, we denote the attack training dataset constructed by the basic and enhanced methods as A^{train} and $A^{\text{train}^\star}$, respectively.

Basic method. The adversary performs the following steps to generate A^{train}: First, N graphs are randomly sampled from the shadow graph G^S. Additionally, a set of node pairs $S = \{(v_j, v_k)\}$ is selected from G^S. A node pair $(v_j, v_k) \in S$ is classified as a member node pair if it is connected in the shadow graph G^S, otherwise, it is classified as a non-member node pair. To ensure a balanced distribution of member and non-member samples in A^{train}, S has the same number of member and non-member edges.

Second, for each selected node pair $(v_j, v_k) \in S$, the adversary computes the similarity between their embeddings in the augmented view $\hat{G}^i$, denoted as $sim(z^i_j, z^i_k)$. Multiple similarity metrics can be used to evaluate the similarity between node embeddings. In this paper, we consider two widely-used similarity functions (i.e., $t = 2$): Cosine similarity and Dot Product.

Third, for each node pair $(v_j, v_k) \in S$, the adversary constructs its concatenated similarity value in the augmented view $\hat{G}^i$ (denoted as $C^i(v_j, v_k)$) as follows:

$$C^i(v_j, v_k) = sim_1(z^i_j, z^i_k)|| \ldots ||sim_t(z^i_j, z^i_k) \tag{3}$$

By concatenating t similarity values, $C^i(v_j, v_k)$ is of length t.

Next, for each selected node pair $(v_j, v_k) \in S$, the adversary assigns all the augmented views into two groups, P^+ and P^-, based on the membership of (v_j, v_k) in these views. Specifically, the augmented views in which (v_j, v_k) is a true positive member are assigned to P^+, and those where (v_j, v_k) is true negative member are assigned to P^-.

Finally, for each member node pair $(v_j, v_k) \in P^+$, the adversary generates its attack feature x by concatenating its similarity values of all of its true positive members across different augmented views:

$$x = ||_{\forall \hat{G}^i \in P^+} C^i(v_j, v_k), \tag{4}$$

And the adversary inserts a data sample (x, "1") into A^{train} for the node pair (v_j, v_k).

Similarly, for each non-member node pair $(v_j, v_k) \in P^-$, the adversary generates its attack feature x as follows:

$$x = ||_{\forall \hat{G}^i \in P^-} C^i(v_j, v_k), \tag{5}$$

where $C^i(v_j, v_k)$ follows Eqn. (3). The adversary then inserts a data sample (x, "0") into A^{train}.

As the vector length of x is determined by either $|P^+|$ or $|P^-|$, it varies for different node pairs. This poses a potential problem for training of the attack model as the feature vector length can vary across different samples. To address this problem, we align the length of the feature vector of all samples in A^{train} to ensure consistency. To explain the details, we first define the *minimum size* ℓ as follows:

$$\ell_{min} = min_{\forall (v_j, v_k) \in A^{\text{train}}}(|P^+|, |P^-|), \tag{6}$$

where $|P^+|$ and $|P^-|$ are the number of augmented views in P^+ and P^-, respectively. The value of ℓ_{min} is determined by the augmentation ratio γ. More discussions of ℓ_{min} and how its value is impacted by different graph augmentation methods will be provided later in this section. After computing ℓ_{min}, the adversary randomly samples ℓ_{min} similarity values from P^+ and P^- respectively for each node pair (v_j, v_k), and generates x (Eqns. (4)) and (5)) from the sampled similarity values. This alignment ensures that the length of the feature vector is consistent across all samples, which is equal to $t \times \ell_{min}$, where t represents the number of similarity metrics.

Enhanced method. As GCL aims to maximize the agreement between any pair of $G^{1,i}$ and $G^{2,i}$ of the same input graph G^i, it is expected that node pairs with the same augmented membership in both $G^{1,i}$ and $G^{2,i}$ should exhibit higher similarity compared to those with different augmented membership. This introduces a *contrastive constraint* on the node pairs in the augmented views, where only node pairs with the same augmented membership in the two contrastive augmented views should be considered for extraction of attack features.

We incorporate the contrastive constraint with the procedure of generating the attack training data and consequently construct $A^{\text{train}^\star}$. The construction of $A^{\text{train}^\star}$ follows a similar approach to A^{train}, with the only difference being that training samples are generated only from node pairs with consistent membership across both contrastive views generated from the same graph sample. Specifically, for each sampled node pair (v_j, v_k), the adversary generates a corresponding training sample only when (v_j, v_k) is a

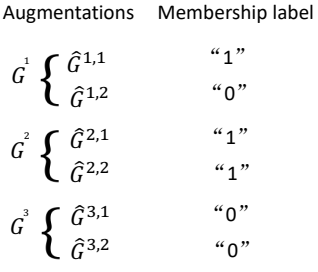


Figure 4: An example of the membership of a node pair (v_j, v_k) in three augmentations. The pair (v_j, v_k) is a member edge in the original training graph.

consistent true positive/negative member in any input graph sample G^i. The ground-truth membership of (v_j, v_k) is still used as the membership label in $A^{\text{train}^\star}$. The feature alignment process remains the same as in A^{train}.

To illustrate the construction of A^{train} and $A^{\text{train}^\star}$, let's consider a GCL model that samples three input graphs G^1, G^2, G^3 from the training graph and constructs six augmented views from these samples. We focus on a node pair (v_j, v_k) that is an edge in the training graph (i.e., it is a member). Figure 4 depicts the membership of (v_j, v_k) in the six augmented views. The following data samples are constructed by GCL-Leak for (v_j, v_k) in A^{train} and $A^{\text{train}^\star}$:

- $A^{\textbf{train}}$: there is one member sample for (v_j, v_k):

$$(C^{1,1}(v_j, v_k)||C^{2,1}(v_j, v_k)||C^{2,2}(v_j, v_k), \text{"1"}).$$

  There is no non-member sample for (v_j, v_k).
- $A^{\textbf{train}^\star}$: there is one member sample for (v_j, v_k):

$$(C^{2,1}(v_j, v_k)||C^{2,2}(v_j, v_k), \text{"1"})$$

  There is no non-member sample for (v_j, v_k).

$A^{\textbf{train}}$ versus $A^{\textbf{train}^\star}$. A^{train} vs $A^{\text{train}^\star}$ exhibit variations in the length of the feature vector. Moreover, different graph augmentation methods can impact the distribution of member and non-member samples in A^{train} and $A^{\text{train}^\star}$. We explain the details below.

$A^{\textbf{train}}$: Recall that A^{train} only considers the similarity values of true positive/negative members. When edge removal is used as the graph augmentation method with an augmentation ratio of γ, each member edge will have one corresponding data sample with an expected feature vector length of $N(1-\gamma)$, labeled as "1". On the other hand, all non-member node pairs will continue to be non-members in all augmented views. Hence, each non-member node pair will have one corresponding data sample with a feature vector length of N. The expected length of the feature vector after alignment is $t \times \ell_{min}$, where

$$\ell_{min} = min\left(N(1-\gamma), N\right) = N(1-\gamma)\,. \tag{7}$$

Furthermore, as A^{train} exclusively contains true positive and negative members, it includes a total of s data samples, with $s/2$ as members and $s/2$ as non-members.

When edge addition is used as the graph augmentation method, A^{train} retains the same expected feature vector length $(N(1-\gamma))$ and sample size (s), following a similar reasoning as above. Due to space constraints, we omit the detailed explanation.

$A^{\textbf{train}^\star}$: When edge removal is used as the graph augmentation method, the probability that a true positive member edge is consistent in both augmented views of the same input graph is $(1-\gamma)^2$. Hence, the expected feature vector length of member samples is $(1-\gamma)^2$, while the length of the feature vector for all non-member samples is N. Therefore, the expected length of the feature vector after alignment is $t\ell$, where

$$\ell_{min} = min\left(N(1-\gamma)^2, N\right) = N(1-\gamma)^2. \tag{8}$$

Similar to A^{train}, $A^{\text{train}^\star}$ comprises s data samples ($s/2$ members and $s/2$ non-members). When edge addition is used as the graph augmentation method, $A^{\text{train}^\star}$ maintains the same expected feature vector length $(N(1-\gamma)^2)$ and the same number of samples (s) as when edge removal is used for augmentation.

5.3 Phase 2: Attack Model Training

Once the training data is generated, the adversary proceeds to train a binary classifier $\mathcal{A}$ for edge membership prediction. Several options for binary classifiers are available, such as Multi-layer Perceptron (MLP), Random Forest (RF), and Linear Regression (LR). In our experiments, we consistently observe that the MLP attack achieves the highest attack accuracy compared to MLP, RF, and LR in the majority of settings. Therefore, we employ an MLP as the attack classifier. The output of the MLP classifier is a probability vector that indicates the likelihood of an instance being a member or non-member. Additional details regarding the setup of the MLP classifier can be found in Section 6.

Unfortunately, as most of the existing LUPI solutions [56, 61, 62] are only applicable to the convex optimization problems, none of them can be applied to our setting. Thus we take a different approach by encoding the identical-membership information within the structure of the attack classifier. Specifically, we utilize $A^{\text{train}^\star}$ to extract attack features so that the attack classifier learns from the embeddings of node pairs with identical membership exclusively. This enables the attack classifier to infer the membership of the node pairs in the testing data even though these pairs do not have such information of identical-membership. We will show that GCL-Leak outperforms the attack models that utilize other LUPI approaches [6, 61] in terms of the attack accuracy (Section 6.5).

5.4 Phase 3: Membership Inference

During the inference stage, for each target pair (v_j, v_k), the adversary extracts its attack feature from the embeddings in all augmented views and inserts a corresponding sample into the attack testing dataset A^{test}. Since the adversary does not possess knowledge about whether (v_j, v_k) is a true positive/negative member in the target graph, ℓ augmented views are randomly selected, where ℓ represents the length of the attack feature. Subsequently, the attack feature of (v_j, v_k) is extracted by concatenating its similarity values in these selected views. As the information of identical-membership is not available, the attack feature is constructed by following Eqn. (4) and (5). This ensures that the MIA testing data maintains the same feature size as the MIA training data. Finally, the adversary inputs A^{test} into the attack classifier $\mathcal{A}$ and obtains a probability vector for all node pairs in the testing data. For each node pair, the class with the higher probability is selected as the predicted class.

5.5 Extension to Node Augmentation based GCL Models

Similar to edge augmentation methods, node augmentations in GCL models can involve either removing nodes along with their edges from augmented views or introducing synthetic nodes as new entities in these views [15, 71, 72, 74]. In cases where node insertion is employed for augmentation, the membership status of all node pairs in the initial training graph remains unchanged in the augmented views. In simpler terms, the augmented membership and ground-truth membership consistently align for each node pair. Consequently, the adversary can straightforwardly leverage the similarity of node pairs in the augmented views to infer their membership in the original graph. Conversely, when node deletion is applied for augmentation, the membership of each node pair becomes inconsistent across different augmented views, mirroring the challenges posed by edge augmentation. As a result, our GCL-Leak can be readily adapted to infer edge membership for GCL models undergoing these forms of node augmentation.

6 PERFORMANCE EVALUATION

We perform an extensive set of experiments, aiming to answer the following three questions: ($\mathbf{Q_1}$): how does GCL-Leak perform against the representative GCL models and real-world datasets; ($\mathbf{Q_2}$): How do various augmentation configurations impact the performance of GCL-Leak? ($\mathbf{Q_3}$): How do various graph properties impact the performance of GCL-Leak? and ($\mathbf{Q_4}$):How does the choice of LUPI techniques impact the performance of GCL-Leak?

6.1 Experimental Setup

All the experiments are executed on a server with NVIDIA A100 GPU and 40GB memory. All the algorithms are implemented in Python along with PyTorch.

Datasets. we utilized three real-world datasets, namely **Cora**, **Citeseer**, and **Facebook**, that are widely used in graph learning research to evaluate the effectiveness of our attack. The statistical information of these datasets can be found in Appendix B.

Target models. We use four state-of-the-art GCL models, namely GRACE [85], MVGRL [14], GCA [86], and CGA-SSG [75]. The details of the parameter setup of these four models and their performance on node classification can be found in Appendix C.

Implementation of attack classifier. We utilize a Multi-layer Perceptron (MLP) consisting of three hidden layers with 64/32/16 neurons at each layer as the attack classifier. ReLU is used as the activation function for the hidden layers, while the Sigmoid function is used for the output layer. The attack classifier is trained for 1,000 epochs with a learning rate of 0.001. We employ cross-entropy loss as the loss function and Adam optimizer.

Training and testing data of attack classifier. To construct the attack training dataset, we partition the original graph's edges into two sets: 70% are randomly assigned as member edges (E_{mem}), and the remaining 30% are designated as non-member edges (E_{non}). The attack training dataset consists of members that are randomly sampled from E_{mem} and non-members that are randomly sampled from E_{non}. The number of members and non-members is the same, which equals $[|E_{non}| \times 0.7]$. The attack testing dataset is generated in a similar way as the training data: it consists of the same number of members and non-members that are randomly sampled from the set of edges that were not sampled for the training data. The testing data is also balanced across members and non-members. The size ratio between the training and testing is 7:3. The actual number of training/testing samples can be found in Appendix D.

Attack evaluation metrics. To assess the attack effectiveness of GCL-Leak, we employ three metrics: (1) *Attack accuracy* –it measures the ratio of the correctly predicted edges (for both members and non-members) by GCL-Leak over the total number of node pairs in the testing data; (2) *Area Under the Curve (AUC)* – it measures AUC of the ROC curve comprising the true positive rate (TPR) and false positive rate (FPR) of the attack classifier at various thresholds; and (3) *True-Positive Rate at False-Positive Rates (TPR@FPR)* [1] – It measures TPR of the attack at various FPR values.

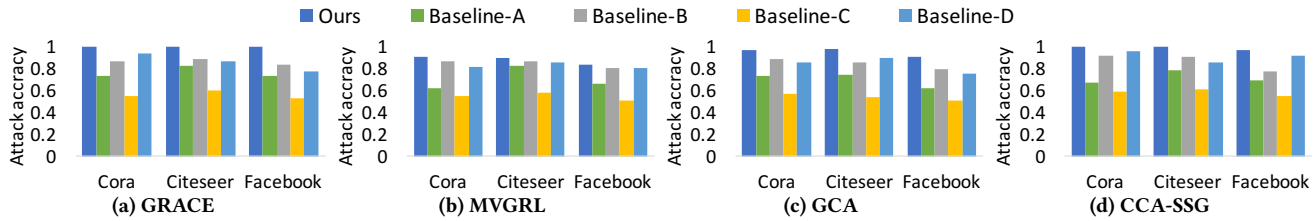
Baseline approaches. We compare GCL-Leak with four baseline methods:

- **Baseline-A (Existing LMIA).** We adapt LMIA [8] to our setting which extracts the attack features from the final node embeddings of the training graph instead of the node embeddings of all the augmented views.
- **Baseline-B (Without LUPI).** By this approach, each data sample in A^{train} corresponds to a node pair, with its attack features extracted by concatenating the similarity values of the node pair in all the augmented views, and its membership label indicating the ground-truth membership of the node pair.
- **Baseline-C (Concatenation of embeddings).** By this method, the attack feature is extracted from the concatenation of embeddings instead of the concatenation of embedding similarity. The ground-truth membership is used as the membership label.
- **Baseline-D (Threshold-base attack).** This method consists of an ensemble of three threshold-based sub-attacks, each using a different similarity metric (Dot product similarity, Cosine similarity, and Euclidean distance). For each sub-attack, a threshold-based LMIA [8] is used to infer the membership of the given node pairs. All node pairs whose embedding similarity is higher than the threshold are considered members, otherwise non-members. The final member/non-member decision is obtained by majority voting over the three attack classifiers.

To ensure a fair comparison between GCL-Leak and the baselines, we ensure the training and testing datasets used by GCL-Leak and the baselines are of the same size.

6.2 Attack Performance (Q_1)

Effectiveness of GCL-Leak. Figure 5 illustrates the attack accuracy of GCL-Leak for different GCL models trained on the three datasets. The shadow graph is sampled from the same graph as the target graph. First, the results demonstrate the high effectiveness of GCL-Leak whose attack accuracy is consistently above 0.83 for all settings, significantly surpassing the random guess accuracy of 0.5. Indeed, GCL-Leak can achieve a perfect attack accuracy of 1 in some settings (e.g., with CCA-SSG model as the target GCL model on Cora and Citeseer datasets, Figure 5 (d)). Second, GCL-Leak outperforms all four baseline approaches in all settings. In particular, GCL-Leak outperforms Baseline-A by a margin (ranging from 0.08 to 0.33). GCL-Leak also notably outperforms Baseline-B on GRACE, GCA, and CCA-SSG models, while achieving only



(a) GRACE (b) MVGRL (c) GCA (d) CCA-SSG

Figure 5: Attack accuracy of GCL-Leak and baseline approaches.

a marginal victory over Baseline-B on the MVGRL model. Similarly, GCL-Leak outperforms Baseline-D by a margin of 0.04 to 0.22. Notably, Baseline-C exhibits the poorest attack performance among the five methods. These results underscore the advantages of GCL-Leak over the baseline methods.

Besides the attack accuracy results, we also measure the TPR@FPR with FPR=0.1% and AUC performance of our GCL-Leak and the four baselines. The observations from these results are consistent with the attack accuracy results (Figure 5): GCL-Leak outperforms the three baselines in all the settings. We include these results in Appendix E due to limited space.

GCL-Leak is effective under transfer setting. We consider the transfer setting where the shadow graph and target graph are sampled from different data distributions and domains. We evaluate the attack performance of GCL-Leak under the transfer setting, and present the results in Figure 6. Our main observations are as follows. First, GCL-Leak remains effective in the transfer setting, with the attack accuracy higher than 0.5 (random guess). In certain settings, the attack accuracy reaches as high as 0.8 (e.g., when Cora is the target graph and Citeseer dataset is the shadow graph, Figure 6 (a)). This demonstrates that GCL-Leak has successfully learned the knowledge of how to distinguish between members and non-members from the shadow graph and effectively transfers this knowledge for the inference attack against the target graph. Furthermore, we observe that GCL-Leak achieves higher effectiveness in transferring knowledge across datasets from the same domain compared to those from different domains. For instance, when the target and shadow datasets are sampled from Cora and Citeseer datasets (both are citation graphs) respectively (Figure 6 (a)), the attack accuracy can reach as high as 0.8. However, when the target dataset is sampled from the Facebook dataset (a social graph) while retaining the shadow dataset sampled from the Citeseer dataset (a citation graph), the attack accuracy drops to 0.61.

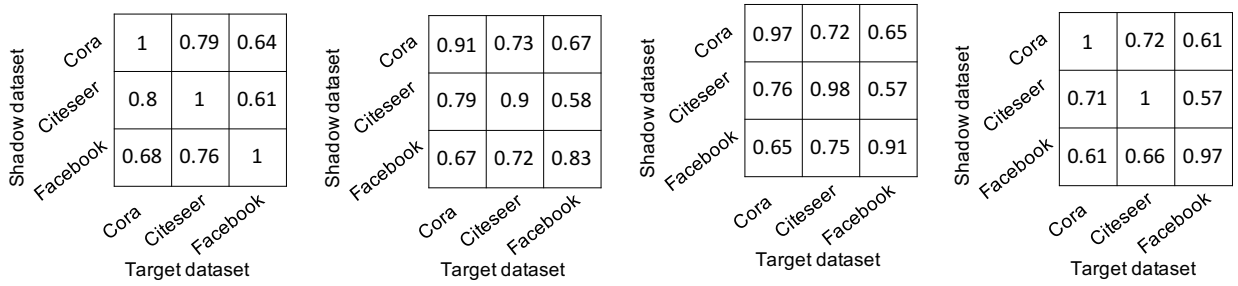
Impacts of similarity metrics on GCL-Leak. We consider various combinations of the three similarity metrics (Cosine similarity, Doc product, and Euclidean distance). Additionally, we consider four distinct aggregation methods (concatenation, average, max pooling, and min pooling) to integrate the similarity values computed by different metrics. The performance of GCL-Leak is evaluated across these diverse settings, and the results are detailed in Appendix E.3. Our analysis reveals two significant findings. Firstly, attacks employing a combination of similarity metrics outperform those relying solely on a single similarity metric. This improvement is attributed to the enhanced classification capabilities of MLP classifiers, leveraging complementary information from different similarity metrics. Secondly, among the four aggregation methods studied, attacks utilizing the concatenation of combined similarity metrics demonstrate superior performance. Consequently, in subsequent discussions, we exclusively consider the setting of GCL-Leak where the attack features are derived from the concatenation of Cosine similarity and Dot Product.

6.3 Impact of Augmentation Configurations (Q_2)

In this section, we explore the impact of four types of GCL augmentation configurations on attack accuracy, namely, augmentation ratio, number of augmented views, edge augmentation methods (edge drop vs. edge addition) and contrastive objectives, on the performance of GCL-Leak. Beyond these configurations, we also evaluate the impact of feature augmentation on attack performance, and found that feature augmentation does not significantly impact the attack accuracy. Thus we include the results of the feature augmentation in Appendix F.

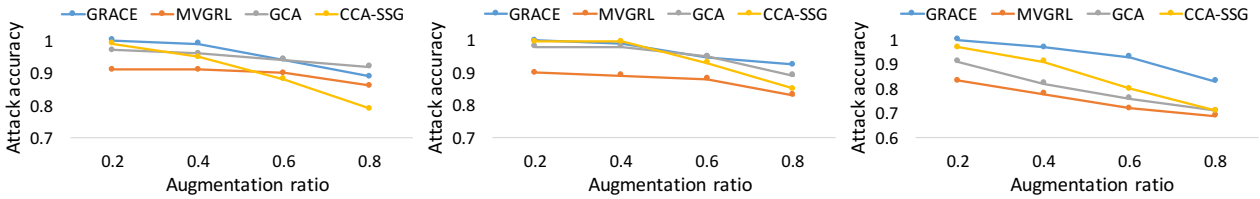
GCL model with a lower augmentation ratio is more vulnerable. We vary the *augmentation ratio*, which represents the percentage of the perturbed edges to the total number of edges in the original graph, to train the GCL models. Then we measure the attack accuracy of GCL-Leak against these models, and present the results in Figure 7. We make the following key observations: (i) In most settings, a higher augmentation ratio consistently leads to lower attack accuracy. This trend holds for all four models, irrespective of their augmentation methods. The observation aligns with our intuition, as a higher augmentation ratio results in a larger number of edges with changed membership across augmented views. Consequently, the feature size of samples in the attack training dataset decreases (Section 5), leading to an expected degradation in attack accuracy. (ii) The attack accuracy exhibits the most significant change on the Facebook dataset compared to the other two datasets as the augmentation ratio increases. We attribute this to the lower homophily of the Facebook dataset compared to the other datasets. Additional details on how graph homophily impacts attack accuracy can be found in Section 6.4. (iii) The attack accuracy demonstrates a more pronounced change for the CCA-SSG model compared to the other models as the augmentation ratio varies. We believe this is partially connected with the contrastive loss functions of the GCL models (more details on the impact of contrastive loss on attack accuracy are discussed later in this section).

GCL model with more augmented views is more vulnerable. Figure 8 presents the attack performance for the GCL models with various numbers of augmented views. As the results indicate, all three datasets exhibit a noticeable increase in attack accuracy with a greater number of augmented views. This observation is easily explained since the number of augmented views determines the size of the attack features (Section 5). With more augmented views, the information encoded in the attack feature increases, resulting in higher attack accuracy.



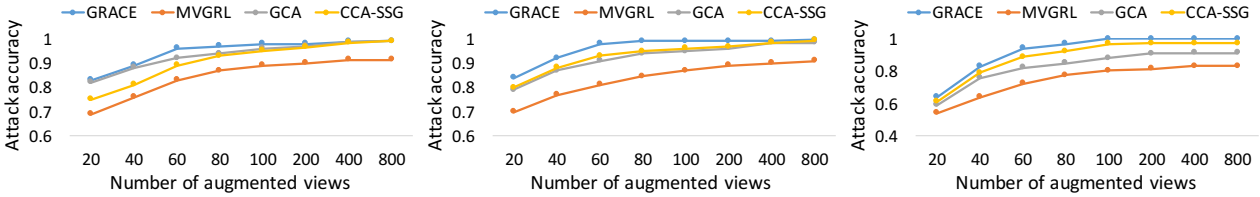
(a) GRACE (b) MVGRL (c) GCA (d) CCA-SSG

Figure 6: Attack accuracy of GCL-Leak under transfer setting.



(a) Cora (b) Citeseer (c) Facebook

Figure 7: Impact of augmentation ratio on attack accuracy



(a) Cora (b) Citeseer (c) Facebook

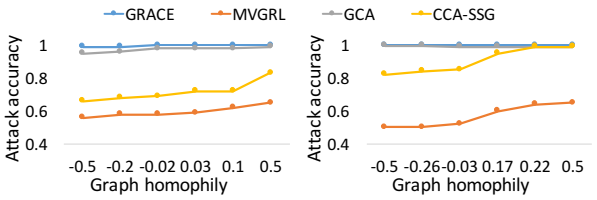
Figure 8: Impact of number of augmented views on attack accuracy.

Edge removal augmentation has a higher impact on privacy vulnerability than edge addition augmentation. As summarized in Table 1, the four GCL models employ different edge augmentation methods, which leads to varying degrees of privacy vulnerability to GCL-Leak. As shown in both Figures 7 and 8, MVGRL consistently exhibits the lowest attack accuracy among the four models in most settings. We suspect this is because the edge addition augmentation method used by MVGRL adds edges that are indirectly linked [7]. Since MVGRL utilizes a 2-layer GCN as its encoder, the node embeddings aggregate information beyond the 1-hop neighbors. Consequently, adding an edge between two indirectly connected nodes results in less change in the embeddings of these nodes compared to removing the edge between them. Thus, edge addition augmentation has a smaller impact on embedding similarity and incurs fewer privacy risks compared with edge removal augmentation. Furthermore, as observed from both Figures 7 and 8, among the three GCL models (GRACE, GCA, CCA-SSG) that employ edge removal for augmentation, GCA consistently demonstrates the lowest attack accuracy in most settings. We believe this is because GCA selectively removes the least important edges, while GRACE and CCA-SSG remove edges randomly. As a result, GCA's augmentation has a lesser impact on embedding similarity in the augmented views compared to the other two models, resulting in lower vulnerability to privacy attacks.

Node-level contrastive objective has less impact on privacy vulnerability than the objective at both node-level and feature-level. As summarized in Table 1, the four GCL models adopt different contrastive objectives. Specifically, GRACE, MVGRL, and GCA only consider node-level contrastive loss, while CCA-SSG incorporates contrastive loss at both node and feature levels. Since all the models include feature augmentation, CCA-SSG is more significantly affected by the augmentation compared to the other three models. As a result, CCA-SSG exhibits the highest variation in attack accuracy when the augmentation ratio changes.

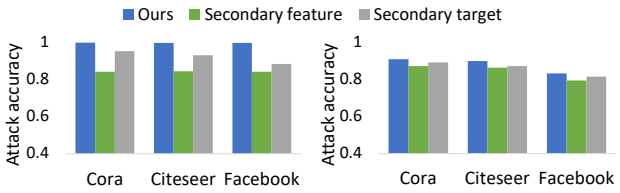
6.4 Impacts of Graph Properties (Q_3)

Graph properties can have an impact on the performance of GCL models. Previous research [72, 84] has identified three graph properties that affect GCL model performance: *graph density*, *graph type*, and *graph homophily*. In this section, we focus on investigating the impact of these properties on attack performance. Our empirical results reveal that graph density and graph type do not significantly impact attack accuracy. Thus we include the results of graph density and graph type in Appendix G, and only present the results of graph homophily in this section.



(a) Facebook (b) Google+

Figure 9: Impact of graph homophily on attack performance.



(a) GRACE (b) MVGRL

Figure 10: Attack accuracy of GCL-Leak under various LUPI paradigms.

At its core, graph homophily refers to the tendency of nodes to connect with others that share similarities [35]. Following the approach of [39], we measure graph homophily using *network modularity*. Formally, given a graph G, suppose all nodes in G are divided into two groups (Groups 0 & 1). Let $c_i = 1$ if a node v_i belongs to Group 1 and $c_i = -1$ otherwise. Then the *network modularity* Q of G is measured as follows:

$$Q = \frac{1}{2m} \sum_{v_i, v_j} (A_{i,j} - \frac{d_i d_j}{2m}) c_i c_j. \tag{9}$$

where m represents the number of edges in G, $A_{i,j}$ denotes the number of edges between nodes v_i and v_j (typically 0 or 1), and d_i and d_j represent the degrees of nodes v_i and v_j, respectively. Generally, Q falls within the range of [-1, 1]. A higher value of Q indicates a higher likelihood of nodes within the same group being connected compared to nodes from different groups.

For this part, we use two new graphs, *Facebook Ego graph*[4] and *Google+ graph*[5] [27]. The statistics information of these two graphs can be found in Appendix B. For each graph, we generate five graph samples of various network modularity by randomly adding/deleting edges. Then we measure the performance of GCL-Leak on the GCL models trained on these graph samples.

Figure 9 presents the attack performance of GCL-Leak on the graphs with varying modularity. We make the following observations. First, higher homophily generally results in higher attack accuracy across most settings. For instance, the attack performance for CCA-SSG is only 0.66 when the homophily of the Facebook Ego graph is -0.5. However, the attack accuracy increases to 0.83 when the graph homophily grows to 0.5. Second, graph modularity affects different GCL models to varying degrees. Specifically, MVGRL and CCA-SSG demonstrate more pronounced changes in attack accuracy in response to changes in graph homophily compared to GRACE and GCA. Next, we explain these two observations.

Why graph modularity can affect attack performance? Intuitively, in the graphs with higher homophily, the connected nodes are more likely to exhibit similarities. As a result, these graphs are less sensitive to edge augmentation compared to graphs with lower homophily. Consequently, the embedding similarity between member and non-member edges becomes more distinguishable in higher homophily graphs, leading to higher attack accuracy.

Why GRACE and GCA are less sensitive than MVGRL and CCA-SSG? We believe this is partly because the extent of change in graph homophily caused by edge addition augmentation in MVGRL is smaller compared to the edge removal augmentation employed by GRACE and GCA. Consequently, the effect of graph homophily on the attack performance is less pronounced in MVGRL. In addition, CCA-SSG utilizes a node-level contrastive objective, which places greater emphasis on node features compared to GRACE and GCA. Thus the impact of edge augmentation on the attack performance is diminished in CCA-SSG. Furthermore, it is possible that the attack accuracy for GRACE and GCA is already sufficiently high even when the graph homophily is as low as -0.5. This suggests that these models are inherently more robust to variations in graph homophily, leading to a reduced sensitivity of GCL-Leak to changes in this graph property.

6.5 Impacts of Various LUPI Methods (Q_4)

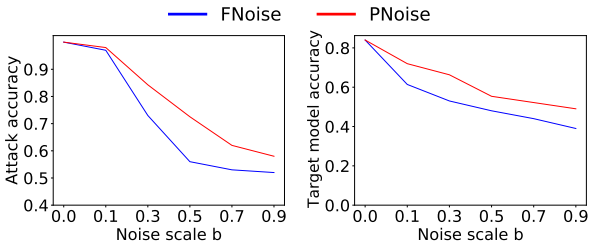
LUPI encompasses various learning strategies, each with distinct methods for handling privileged information [3, 56, 61, 62]. These methods typically involve the representation of privileged information through slack functions [56], regularization terms or a secondary feature [62, 69], and secondary targets [61]. Given that slack functions are commonly employed in Support Vector Machine (SVM)-based models, they cannot be directly applied to the neural network based models such as GCL. Therefore, we mainly consider the other two categories of LUPI approaches. Specifically, we adapt two LUPI algorithms [6, 61] to our problem setting:

Secondary feature [61]: The basic idea of [61] is to represent the privileged information as a secondary feature. The learning model encompasses training a secondary classifier with the secondary feature, whose output is introduced as the regularizer term to the loss function of the primary classifier. To adapt this method to our setting, we treat the augmented membership of each node pair as a secondary feature. More details of how to add the secondary feature to our attack model can be found in Appendix H.

Secondary target [6]: Basically, [6] treats the privileged information as the secondary target. The learning model's objective is to minimize the loss associated with both the primary target and the secondary target. To adapt [6] to our setting, we treat the attack model as a multi-learning task. The objective of this multi-learning task is to include the loss of membership discrimination (evaluated over the ground-truth memberships) and the loss from the secondary target (evaluated over the memberships in the augmented views). More details on how to add the secondary feature to our attack model can be found in Appendix H.

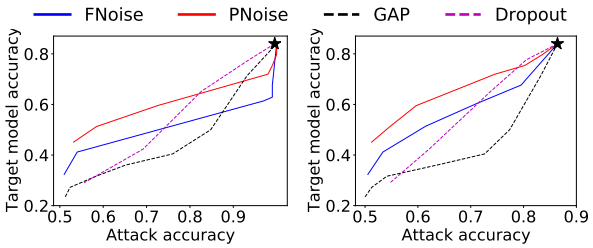
Comparison between different LUPI approaches. Figure 10 presents the accuracy of the attack models equipped with the three different LUPI methods. The main observation is that, although the three methods have comparable attack accuracy, GCL-Leak outperforms the other two methods across all the settings. In particular, the attack accuracy of GCL-Leak can be as high as 1, while

[4]Facebook Ego graph is a social network graph sampled from Facebook dataset [27]. It is different from the Facebook dataset that we used in the previous experiments as it has higher network modularity.
[5]https://snap.stanford.edu/data/ego-Gplus.html



(a) Attack performance (b) Target model accuracy

Figure 11: FNoise vs. PNoise (GRACE model, Cora dataset, $r = 0.4$ for PNoise). $b = 0$ indicates no defense.



(a) GCL-Leak as attack (b) W-LMIA [8] as attack

Figure 12: Defense-utility trade-off of our defense methods and the two baseline methods against both GCL-Leak and W-LMIA [8] attacks (GRACE model, Cora dataset, $r = 0.4$ for PNoise). The ⋆ symbol indicates no defense.

the attack accuracy of the other two methods is 0.87 and 0.95 respectively. This demonstrates the effectiveness of GCL-Leak in effectively handling the "identical-membership" information.

7 POSSIBLE DEFENSES

Given the effectiveness of GCL-Leak against GCL models, in this section, we design countermeasures to mitigate the privacy risk of GCL models.

Details of defense mechanisms. Intuitively, as GCL-Leak utilizes the node embeddings as the input, adding perturbations to node embeddings can diminishing the attacker's ability to infer edge membership. In this paper, we follow the basic idea of [48] and incorporate noise directly to the node embeddings. Specifically, for each augmented view $\hat{G}$ and each node $v_i \in \hat{G}$, we add noise to its embedding z_i. The noise follows the Laplace distribution of the density function $\frac{1}{2b}e^{-\frac{x-\mu}{b}}$ where b is the noise scale and μ is the location parameter of the Laplace distribution. Intuitively, a higher value of b indicates more noise added to the embedding. On the other hand, adding more noise can lead to higher accuracy loss of the GCL models. To address the trade-off between privacy and model accuracy, we consider two different methods to inject noise:

- **Full noise (FNoise)**: Given a node embedding z, the noise is injected to all the dimensions of z.
- **Partial noise (PNoise)**: Given a node embedding z, the noise is injected selectively only to the dimensions of z that are least important to the target model. For the node embeddings from each augmentation, we evaluate the importance of each embedding dimension in terms of the effectiveness of node classification, which is measured by the SHapley Additive exPlanations (SHAP) method [32]. We sort the dimensions by their SHAP values, and perturb $[d \times r]$ dimensions of the lowest importance, where d is the total number of dimensions, and $r \in (0, 1)$ is the perturbation ratio. A higher perturbation ratio leads to more dimensions to be perturbed. When $r = 1$, it is equivalent to FNoise.

Unfortunately, as both FNoise and PNoise do not utilize the concept of adjacency of datasets to determine the amounts of noise, they cannot offer a formal guarantee of differential privacy (DP) [9]. However, our empirical evaluation will demonstrate that both FNoise and Noise outperform a DP-based GNN [44] in terms of the trade-off between defense effectiveness and model accuracy.

Baselines. We adapt two existing defense mechanisms to our setting as the baseline approaches: (1) GAP [44]: GAP was originally designed to provide both node-level and edge-level differential privacy for GNNs. We adapted GAP to GCL models [6] to add the Gaussian noise to node embeddings in the augmented views. In our experiments, we follow the same setup of GAP and calibrate the noise scale with the given privacy budget $\epsilon = \{0.01, 0.1, 1, 5, 10\}$. (2) Dropout: We adopt the dropout technique, which has been demonstrated as an effective defense against MIA for the non-graph classification models [45]. We adapted this technique to GCL models by randomly halting the update of α percentage of neurons within the GCL model during each training iteration, where $\alpha \in [0, 100\%)$ is the dropout ratio. In our experiments, we use $\alpha = \{20\%, 40\%, 60\%, 80\%\}$.

Evaluation metrics. To evaluate the effectiveness of the defense mechanism, we measure both attack accuracy and target model accuracy after deploying the defense mechanisms. Regarding attack accuracy, we use the same evaluation metric as defined in Section 6. In terms of target model accuracy, we measure the target model accuracy as the accuracy of node classification: $AC = n_t^*/n_t$, where n_t^* is the number of nodes in the testing data that are correctly classified, and n_t is the total number of nodes in the testing data.

Performance of our defenses. Figure 11 presents the performance of FNoise and PNoise across various noise levels on the GRACE model. First, both FNoise and PNoise are effective against LMIAs by witnessing a decrease in the attack accuracy. The effectiveness of both methods improves with higher noise scales. When the noise scale b reaches 0.9, both methods can reduce the attack accuracy to approximately 0.5. Indeed, there exists a trade-off between defense effectiveness and target model accuracy. As it adds the noise across all dimensions, FNoise is more effective than PNoise to protect against LMIA, but it incurs higher accuracy loss of the target model. The additional results for other target models and datasets can be found in Appendix I.

Comparison with baselines. As our defense methods and the two baselines use different parameters to control the defense power, it is difficult to compare these methods in terms of their attack effectiveness. Thus, we compare these methods in terms of their trade-off between the defense power and target model accuracy. In particular, we generate the *defense-utility curve* in which each point presents a pair of attack accuracy and target model accuracy (utility) values. To generate the curve, we vary the parameter setting of our defense methods and the two baselines, and measure both attack accuracy and target model accuracy for each setting. Intuitively, a

[6] We use the implementation of GAP available at https://github.com/sisaman/GAP

defense method that has lower attack accuracy and higher target model accuracy has a better defense-utility trade-off.

Figure 12 (a) presents the defense-utility curve of our defense mechanisms and the two baselines. We observe that, although both FNoise and PNoise cannot provide a formal privacy guarantee as GAP, they can provide comparable amounts of protection against both attacks as GAP. Furthermore, both FNoise and PNoise outperform the two baselines in the trade-off between defense effectiveness and target model accuracy, especially when the attack accuracy is reduced to be lower than 0.75. Both FNoise and PNoise achieve lower attack accuracy than the baselines under the same target model accuracy, while they deliver higher target model accuracy than the baselines under the same attack accuracy. In addition, PNoise exhibits the best defense-utility trade-off among the four methods. The results for other datasets can be found in Appendix I.

Defense against other attacks. To have a better understanding of the defense power of our methods, besides GCL-Leak, we evaluate the performance of our defense methods against the white-box LMIA attack (**W-LMIA**) [8], whose attack features were derived from the final node embeddings learned from the embeddings of all augmented views. Figure 12 (b) presents the defense-utility curve of our defense mechanisms and the two baselines against LMIA. Our observation is similar to GCL-Leak attack: both FNoise and PNoise are effective against W-LMIA, as they are able to reduce the attack accuracy to around 0.5. Furthermore, both FNoise and PNoise outperform the two baselines in terms of the defense-utility trade-off as the defense strength increases.

Factor analysis of defense-utility trade-off. For a comprehensive understanding of the trade-off between defense effectiveness and model accuracy, we investigate the impact of two graph properties—graph density and graph homophily—along with one specific type of data distribution, namely class distribution, on the defense-utility trade-off. We consider these three factors due to their recognized impact on LMIA performance [17, 83]. We present the results in Appendix G. The key observation is that the graph density exerts a substantial impact on the defense-utility trade-off, as our defense method achieves better defense-utility trade-off on sparse graphs than dense ones. On the other hand, we do not observe any discernible impact of graph homophily and class distribution on the defense-utility trade-off.

8 RELATED WORK

Membership inference attacks and defense. Shokri et al. [49] initialized the research on membership inference attacks (MIAs) against ML models. Yeom et al. [70] studied the relationship between overfitting and MIA in general. Salem et al. [45] relaxed the assumptions of MIA in these initial works. MIA has been designed in both black-box and white-box settings. In the black-box setting, the adversary utilizes either the prediction probabilities (posteriors) [17, 49, 70, 73] or the prediction labels [5, 28], while in the white-box setting, the adversary utilizes the access to the target model parameters [17, 73], the loss [43, 70], and the gradients [37]. Recently, MIAs have been extended to various application scenarios, including generative models [16], language models [50, 51], recommender system [76], and graph neural networks [17, 40].

Numerous defense mechanisms have been proposed to counter MIA, falling into four main categories: (1) Confidence score masking [23, 49] adds noise to the confidence scores generated by the target model. (2) Regularization techniques [16, 19, 28, 36] incorporate a regularization term into the objective function to enhance privacy. (3) Knowledge distillation approaches [47, 82] transfer knowledge from a larger model to a smaller one while minimizing information leakage about data membership. (4) Differential privacy methods [2, 5, 16, 49] offer a rigorous privacy guarantee by quantifying the privacy of individual samples. For an in-depth survey of MIA attacks and defenses, we refer readers to [20].

Privacy attacks against graph learning models. Various types of privacy attacks have been designed recently to attack graph learning models. These attacks can be classified into the following categories: (1) The membership inference attacks (MIA) [8, 17, 18, 48, 63, 79]; (2) the attribute inference attacks (AIA) [8, 12, 77] that aim to infer the sensitive attributes in the training graph; and (3) the property inference attacks (PIA) [54, 60, 78, 80] that try to infer the sensitive property of the training graph. MIA has been investigated at node level [18] (i.e., whether a node is included in the training graph), link level [8, 17, 48, 63, 79], and subgraph level [80], with GNNs as the attack target. While most of the existing link-level MIA share the same goal as ours, none of them can be directly adapted to GCL models, due to the disappearance of the distinguishability of members and non-members by their embedding similarity. Thus we design new LMIA attacks against GCL models.

Privacy attacks against self-supervised learning models. There has been limited research on the privacy vulnerabilities of self-supervised learning models. Liu et al. [29] designed the first membership inference attack against the contrastive learning models. However, [29] only considered the image data. Due to the fundamental difference between augmentation methods on graphs and images, these attacks cannot be launched against the GCL models. Liu et al. [31] proposed an attack to recover the target encoder used in self-supervised learning models. While they focus on model privacy, we focus on training data privacy.

9 CONCLUSION

This paper initializes the study of privacy risks of GCL models through the lens of membership inference attacks. We design GCL-Leak, the first white-box link inference attacks against GCL models. Our experiments demonstrate the effectiveness of GCL-Leak against four representative GCL models. We also design a defense mechanism against GCL-Leak and empirically demonstrate the effectiveness of the defense mechanism.

Future work. This paper only investigated the privacy vulnerability of GCL models against the membership inference attack. Examining the privacy risks of GCL models by other types of attacks such as attribute inference attacks [11, 12, 22] and property inference attacks [10, 38, 67] is one interesting direction to explore. Another direction of future work is to design alternative effective defense mechanisms against GCL-Leak.

ACKNOWLEDGEMENT

We thank the anonymous reviewers for their feedback. This project was supported by the National Science Foundation (#CNS-2029038; #CNS-2135988). Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agency.

REFERENCES

[1] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *Proceedings of the Conference on Symposium on Security and Privacy*, pages 1897–1914, 2022.

[2] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. Gan-leaks: A taxonomy of membership inference attacks against generative models. In *Proceedings of the ACM SIGSAC conference on computer and communications security*, pages 343–362, 2020.

[3] Jixu Chen, Xiaoming Liu, and Siwei Lyu. Boosting with side information. In *Asian Conference on Computer Vision*, pages 563–577, 2013.

[4] Mingyang Chen, Wen Zhang, Zonggang Yuan, Yantao Jia, and Huajun Chen. Federated knowledge graph completion via embedding-contrastive learning. *Knowledge-Based Systems*, 252:109459, 2022.

[5] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks. In *International conference on machine learning*, pages 1964–1974, 2021.

[6] Grzegorz Chrupała, Ákos Kádár, and Afra Alishahi. Learning language through pictures. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, pages 112–118, 2015.

[7] Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. Data augmentation for deep graph learning: A survey. *ACM SIGKDD Explorations Newsletter*, 24(2):61–77, 2022.

[8] Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. Quantifying privacy leakage in graph embedding. In *Proceedings of 17th EAI International Conference on Mobile and Ubiquitous Systems*, pages 76–85, 2020.

[9] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19, 2008.

[10] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the ACM SIGSAC conference on computer and communications security*, pages 619–633, 2018.

[11] Neil Zhenqiang Gong and Bin Liu. You are who you know and how you behave: Attribute inference attacks via users' social friends and behaviors. In *Proceedings of the USENIX Security Symposium*, pages 979–995, 2016.

[12] Neil Zhenqiang Gong and Bin Liu. Attribute inference attacks in online social networks. *ACM Transactions on Privacy and Security (TOPS)*, 21(1):1–30, 2018.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[14] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *Proceedings of the International Conference on Machine Learning*, pages 4116–4126, 2020.

[15] Kaveh Hassani and Amir Hosein Khasahmadi. Learning graph augmentations to learn graph representations. *arXiv preprint arXiv:2201.09830*, 2022.

[16] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. Logan: Membership inference attacks against generative models. *In Proceedings of the Privacy Enhancing Technologies (PoPETs)*, 2017.

[17] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. Stealing links from graph neural networks. In *Proceedings of the 30th USENIX Security Symposium*, pages 2669–2686, 2021.

[18] Xinlei He, Rui Wen, Yixin Wu, Michael Backes, Yun Shen, and Yang Zhang. Node-level membership inference attacks against graph neural networks. *arXiv preprint arXiv:2102.05429*, 2021.

[19] Benjamin Hilprecht, Martin Härterich, and Daniel Bernau. Monte carlo and reconstruction membership inference attacks against generative models. *Proceedings of the Privacy Enhancing Technology*, 2019(4):232–249, 2019.

[20] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)*, 54(11s):1–37, 2022.

[21] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *Conference on Learning Representations (ICLR)*, 2020.

[22] Jinyuan Jia and Neil Zhenqiang Gong. {AttriGuard}: A practical defense against attribute inference attacks via adversarial machine learning. In *Proceedings of the USENIX Security Symposium*, pages 513–529, 2018.

[23] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. Memguard: Defending against black-box membership inference attacks via adversarial examples. In *Proceedings of the ACM SIGSAC conference on computer and communications security*, pages 259–274, 2019.

[24] Wei Jin, Tyler Derr, Haochen Liu, Yiqi Wang, Suhang Wang, Zitao Liu, and Jiliang Tang. Self-supervised learning on graphs: Deep insights and new direction. *arXiv preprint arXiv:2006.10141*, 2020.

[25] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[26] Klas Leino and Matt Fredrikson. Stolen memories: Leveraging model memorization for calibrated white-box membership inference. In *29th USENIX security symposium (USENIX Security)*, pages 1605–1622, 2020.

[27] Jure Leskovec and Julian Mcauley. Learning to discover social circles in ego networks. *Advances in neural information processing systems*, 25, 2012.

[28] Jiacheng Li, Ninghui Li, and Bruno Ribeiro. Membership inference attacks and defenses in classification models. In *Proceedings of the ACM Conference on Data and Application Security and Privacy*, pages 5–16, 2021.

[29] Hongbin Liu, Jinyuan Jia, Wenjie Qu, and Neil Zhenqiang Gong. Encodermi: Membership inference against pre-trained encoders in contrastive learning. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 2081–2095, 2021.

[30] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and Philip Yu. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[31] Yupei Liu, Jinyuan Jia, Hongbin Liu, and Neil Zhenqiang Gong. Stolenencoder: Stealing pre-trained encoders in self-supervised learning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2115–2128, 2022.

[32] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[33] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52, 2015.

[34] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282, 2017.

[35] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, pages 415–444, 2001.

[36] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 634–646, 2018.

[37] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753, 2019.

[38] Muhammad Naveed, Seny Kamara, and Charles V Wright. Inference attacks on property-preserving encrypted databases. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 644–655, 2015.

[39] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[40] Iyiola E Olatunji, Wolfgang Nejdl, and Megha Khosla. Membership inference attack on graph neural networks. In *Proceedings of the 3rd IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 11–20, 2021.

[41] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[42] Dmitry Pechyony and Vladimir Vapnik. On the theory of learnining with privileged information. *Advances in neural information processing systems*, 23, 2010.

[43] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Hervé Jégou. White-box vs black-box: Bayes optimal strategies for membership inference. In *International Conference on Machine Learning*, pages 5558–5567, 2019.

[44] Sina Sajadmanesh, Ali Shahin Shamsabadi, Aurélien Bellet, and Daniel Gatica-Perez. Gap: Differentially private graph neural networks with aggregation perturbation. In *USENIX Security 2023-32nd USENIX Security Symposium*, 2023.

[45] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *Network and Distributed Systems Security (NDSS) Symposium*, 2018.

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

[47] Virat Shejwalkar and Amir Houmansadr. Membership privacy for machine learning models through knowledge transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9549–9557, 2021.

[48] Yun Shen, Yufei Han, Zhikun Zhang, Min Chen, Ting Yu, Michael Backes, Yang Zhang, and Gianluca Stringhini. Finding mnemon: Reviving memories of node embeddings. In *Proceedings of the ACM SIGSAC conference on computer and communications security*, pages 2643–2657, 2022.

[49] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *IEEE symposium on security and privacy (SP)*, pages 3–18, 2017.

[50] Congzheng Song and Ananth Raghunathan. Information leakage in embedding models. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 377–390, 2020.

[51] Congzheng Song and Vitaly Shmatikov. Auditing data provenance in text-generation models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 196–206, 2019.
[52] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *International Conference on Learning Representations*, 2019.
[53] Susheel Suresh, Danny Godbout, Arko Mukherjee, Mayank Shrivastava, Jennifer Neville, and Pan Li. Federated graph representation learning using self-supervision. *arXiv preprint arXiv:2210.15120*, 2022.
[54] Anshuman Suri and David Evans. Formalizing and estimating distribution inference risks. *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*, 2022.
[55] Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, and Jing Jiang. Federated learning from pre-trained models: A contrastive learning approach. *arXiv preprint arXiv:2209.10083*, 2022.
[56] Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural networks*, 22(5-6):544–557, 2009.
[57] Petar Veli**v**cković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *Conference on Learning Representations (ICLR)*, 2018.
[58] Haonan Wang, Jieyu Zhang, Qi Zhu, and Wei Huang. Augmentation-free graph contrastive learning. *arXiv preprint arXiv:2204.04874*, 2022.
[59] Tingqi Wang, Xu Zheng, Lei Gao, Tianqi Wan, and Ling Tian. Sc-fgcl: Self-adaptive cluster-based federal graph contrastive learning. *IEEE Open Journal of the Computer Society*, 2023.
[60] Xiuling Wang and Wendy Hui Wang. Group property inference attacks against graph neural networks. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security*, pages 2871–2884, 2022.
[61] Ziheng Wang and Qiang Ji. Classifier learning with hidden information. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4969–4977, 2015.
[62] Ziheng Wang, Xiaoyang Wang, and Qiang Ji. Learning with hidden information. In *Proceedings of the Conference on Pattern Recognition*, pages 238–243, 2014.
[63] Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. Linkteller: Recovering private edges from graph neural networks via influence analysis. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2022.
[64] Lirong Wu, Haitao Lin, Cheng Tan, Zhangyang Gao, and Stan Z Li. Self-supervised learning on graphs: Contrastive, generative, or predictive. *IEEE Transactions on Knowledge and Data Engineering*, 2021.
[65] Han Xie, Jing Ma, Li Xiong, and Carl Yang. Federated graph classification over non-iid graphs. *Advances in Neural Information Processing Systems*, 34:18839–18852, 2021.
[66] Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. Self-supervised learning of graph neural networks: A unified review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.
[67] Mingxue Xu and Xiangyang Li. Subject property inference attack in collaborative learning. In *Proceedings of the IEEE Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, volume 1, pages 227–231, 2020.
[68] Haoran Yang, Xiangyu Zhao, Muyang Li, Hongxu Chen, and Guandong Xu. Federated graph contrastive learning. *arXiv preprint arXiv:2207.11836*, 2022.
[69] Yazhou Yao, Fumin Shen, Jian Zhang, Li Liu, Zhenmin Tang, and Ling Shao. Extracting privileged information for enhancing classifier learning. *IEEE Transactions on Image Processing*, 28(1):436–450, 2018.
[70] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *IEEE 31st computer security foundations symposium (CSF)*, pages 268–282, 2018.
[71] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. In *International Conference on Machine Learning*, pages 12121–12132. PMLR, 2021.
[72] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823, 2020.
[73] Oualid Zari, Chuan Xu, and Giovanni Neglia. Efficient passive membership inference attack in federated learning. *arXiv preprint arXiv:2111.00430*, 2021.
[74] Jiaqi Zeng and Pengtao Xie. Contrastive self-supervised learning for graph classification. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 35, pages 10824–10832, 2021.
[75] Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, and Philip S Yu. From canonical correlation analysis to self-supervised graph neural networks. *Advances in Neural Information Processing Systems*, 34:76–89, 2021.
[76] Minxing Zhang, Zhaochun Ren, Zihan Wang, Pengjie Ren, Zhunmin Chen, Pengfei Hu, and Yang Zhang. Membership inference attacks against recommender systems. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 864–879, 2021.
[77] Shijie Zhang, Hongzhi Yin, Tong Chen, Zi Huang, Lizhen Cui, and Xiangliang Zhang. Graph embedding for recommendation against attribute inference attacks. In *Proceedings of the Web Conference 2021*, pages 3002–3014, 2021.
[78] Wanrong Zhang, Shruti Tople, and Olga Ohrimenko. Leakage of dataset properties in {Multi-Party} machine learning. In *Proceedings of the USENIX Security Symposium*, pages 2687–2704, 2021.
[79] Zaixi Zhang, Qi Liu, Zhenya Huang, Hao Wang, Chengqiang Lu, Chuanren Liu, and Enhong Chen. Graphmi: Extracting private graph data from graph neural networks. *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.
[80] Zhikun Zhang, Min Chen, Michael Backes, Yun Shen, and Yang Zhang. Inference attacks against graph neural networks. In *Proceedings of the USENIX Security Symposium*, pages 1–18, 2022.
[81] Tong Zhao, Gang Liu, Stephan Günnemann, and Meng Jiang. Graph data augmentation for graph machine learning: A survey. *arXiv preprint arXiv:2202.08871*, 2022.
[82] Junxiang Zheng, Yongzhi Cao, and Hanpin Wang. Resisting membership inference attacks through knowledge distillation. *Neurocomputing*, 452:114–126, 2021.
[83] Da Zhong, Ruotong Yu, Kun Wu, Xiuling Wang, Jun Xu, and Wendy Hui Wang. Disparate vulnerability in link inference attacks against graph neural networks. *Proceedings on Privacy Enhancing Technologies*, 2023.
[84] Yanqiao Zhu, Yichen Xu, Qiang Liu, and Shu Wu. An empirical study of graph contrastive learning. *NeurIPS Datasets and Benchmarks*, 2021.
[85] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *Proceedings of the ICML workshop*, 2020.
[86] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference*, pages 2069–2080, 2021.
[87] Marinka Zitnik, Rok Sosi**v**c, and Jure Leskovec. Prioritizing network communities. *Nature communications*, 9(1):1–9, 2018.

APPENDIX

A DISTRIBUTION OF MEMBERS AND NON-MEMBERS

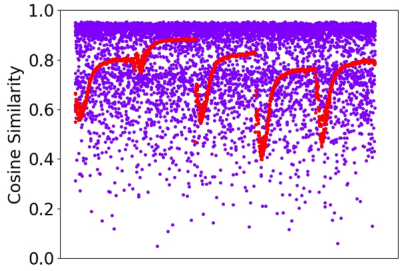
Distribution of augmented/ground-truth members/non-members by embedding similarity in augmented views. Figure 13 visualizes the distribution of members and non-members by the Cosine similarity of their node embeddings when the MVGRL model is used as the target GCL model. The main observations are similar to Section 4 and thus are omitted.

Distribution of ground-truth members/non-members by their attack features. Figure 14 depicts the t-SNE distribution of member and non-member node pairs in the Cora dataset. We observe that the member and non-member links are distinguishable for all four GCL models. This observation explains why GCL-Leak can accurately infer the membership of node pairs.

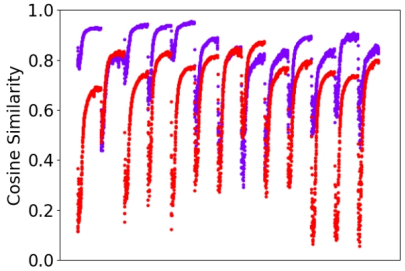
B DESCRIPTION OF DATASETS

We conduct our experiments using five datasets, each with its own characteristics:

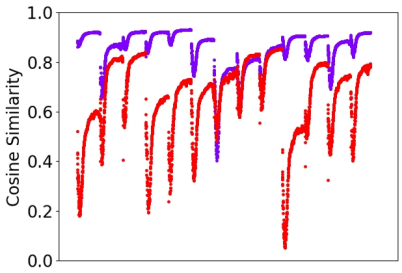
- **Cora** [46] is a citation graph where nodes represent scientific publications and edges represent citation links. It consists of 2,708 publications categorized into seven classes, with a total of 5,429 links.
- **Citeseer** [46] is a citation graph where nodes represent documents and edges represent citations. It contains 3,312 nodes classified into six classes, with 4,732 citation links.
- **Facebook** [27] is a social graph consisting of Facebook users as nodes and their friendship relationships as edges. It comprises 4,039 nodes and 88,234 edges.
- **Facebook - Ego** [27] is a subset of the Facebook social graph, sampled from the larger dataset. It includes 1,050 nodes and 24,191 edges.



(a) Distribution of augmented membership labels by node pairs' embedding similarity in augmented views.

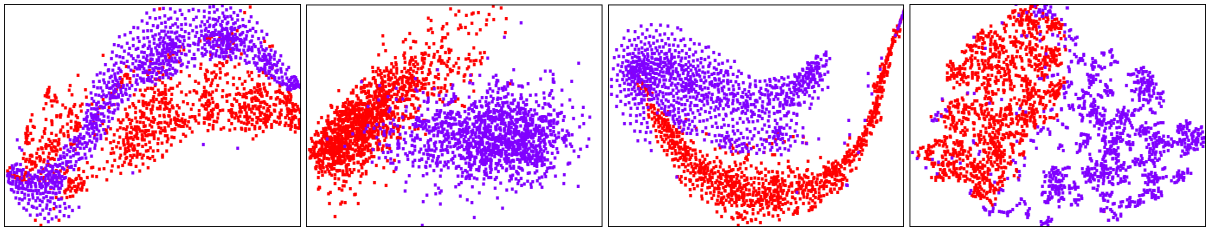


(b) Distribution of ground-truth membership labels by Cosine similarity of node embeddings in augmented views.



(c) Distribution of ground-truth membership labels by Cosine similarity of embeddings of nodes with consistent membership in augmented views and training graph.

Figure 13: Distribution of augmented/ground-truth members/non-members by embedding similarity in augmented views (Cora dataset, MVGRL model). Members and non-members are denoted in blue and red respectively. X-axis: Member and non-member samples, y-axis: Cosine similarity between node embeddings.



(a) GRACE (b) MVGRL (c) GCA (d) CCA-SSG

Figure 14: Distribution of MIA input features of $A^{\text{train}^\star}$ for four GCL models. embers and non-members are denoted in blue and red respectively.

Dataset	$\|V\|$	$\|E\|$	$\|C\|$	Density	Modularity
Cora	2,708	5,429	6	0.0014	0.29
Citeseer	3,312	4,732	7	0.0006	0.36
Facebook	4,039	88,234	2	0.011	0.09
Facebook - Ego	1,050	24,191	2	0.044	0.1
Google+	4,417	119,582	2	0.012	0.22

Table 2: Description of datasets ($|V|$: number of nodes, $|E|$: number of edges, $|C|$: number of of classes).

Dataset	GRACE	MVGRL	GCA	CCA-SSG
Cora	0.84	0.84	0.83	0.84
Citeseer	0.72	0.74	0.75	0.73
Facebook	0.72	0.73	0.73	0.73

Table 3: Performance of the four GCL models.

- **Google+** [27] is a social graph comprising 4,417 nodes and 119,582 edges. Each node is associated with features such as gender, institution, job title, last name, place, and university.

Table 2 presents the statistics of these five datasets.

C SETUP AND PERFORMANCE OF TARGET GCL MODELS

We adhere to the parameter setup described in the publications of the four GCL models: GRACE [85], MVGRL [14], GCA [86], and CCA-SSG [75]. Specifically, we utilize the same encoder model, which is a 2-layer Graph Convolutional Network (GCN), for all four GCL models. The embedding sizes for Cora, Citeseer, and Facebook are set as follows: 128/256/128 for GRACE, 512/512/512 for MVGRL and CCA-SSG, and 256/256/256 for GCA.

For the downstream task of the GCL models, we consider node classification and use the accuracy of node classification as the performance metric. The results of node classification performance for these GCL models are presented in Table 3. We compare the GCL models' performance with different augmentation approaches (node degree, eigenvector, and pagerank centrality-based), and observe that the highest performance is achieved using pagerank centrality-based augmentation for Cora, Citeseer, and Facebook datasets, respectively. Hence, we employ the augmentation approach that yields the highest node classification performance for performing MIA on different datasets. Since the accuracy of all three models is significantly higher than the random guess, these models are susceptible to MIA.

Dataset	Attack Training Data		Attack Testing Data	
	Members	Non-mem	Members	Non-mem
Cora	3,800	3,800	1,628	1,628
Citeseer	3,312	3,312	1,419	1,419
Facebook	61,763	61,763	26,470	26,470
Facebook - Ego	16,933	16,933	7,257	7,257
Google+	83,707	83,706	35,874	35,874

Table 4: Number of training/testing samples for the attack classifier (Non-mem represents Non-members).

D DETAILS OF ATTACK TRAINING AND TESTING DATA

Table 4 presents the number of training/testing samples for the attack classifier. We ensure balanced data in both the training and testing phases of the attack.

E ADDITIONAL PERFORMANCE RESULTS OF GCL-LEAK

E.1 TPR@FPR Performance of Attacks

For this part, we do not consider Baseline-D as it is not a supervised LMIA attack and thus cannot be evaluated with TPR and FPR. Table 5 shows the TPR@0.1%FPR performance of our attack and the three baseline approaches. Overall, GCL-Leak outperforms the three baselines in all the settings in terms of TPR@0.1% FPR.

E.2 AUC Performance of Attacks

Table 6 shows the AUC performance of GCL-Leak. We observe that the attack AUC of GCL-Leak ranges in [0.84, 1], significantly surpassing the value of 0.5 (random guess). In certain settings, the AUC approaches 1, indicating the effectiveness of GCL-Leak against the GCL models.

E.3 Attack Performance of Various Combinations and Aggregations of Similarity Metrics

In this part of experiments, we investigate the impact of different combinations of similarity metrics as well as various aggregation methods of these similarity metrics on the performance of GCL-Leak.

Various combinations of similarity metrics. In Table 7, we present the attack accuracy of GCL-Leak under different combinations of similarity metrics. The results show that the attacks that leverage multiple similarity metrics outperform those reliant on a single similarity metric. Specifically, the attack that utilizes all the three similarity metrics exhibits the best attack performance. Furthermore, the attack that utilizes the combination of Cosine similarity and Dot Product exhibits close attack performance as the one that uses the three similarity functions. This suggests a degree of redundancy in these combinations with regard to their effectiveness.

Various aggregation methods. To better understand the impact of the different similarity aggregation methods on attack performance, we consider four distinct aggregation methods, namely concatenation, average, min-pooling, and max-pooling. As shown in Table 8, the attack employing similarity concatenation outperforms the ones that employ the other three aggregation methods. This superiority can be attributed to the fact that the concatenation method preserves a more comprehensive set of information regarding the similarity between node embeddings.

Given these observations, we opt to employ the concatenation of Cosine similarity and Dot Product as our chosen attack input feature.

F IMPACT OF FEATURE AUGMENTATION ON ATTACK PERFORMANCE

Table 9 presents the results of attack performance on GCL models with and without feature augmentation. We observe that the attack accuracy of the four GCL models remains stable regardless of the usage of feature augmentation. This suggests that feature augmentation has little impact on the attack performance.

G ADDITIONAL RESULTS OF IMPACT OF GRAPH PROPERTIES

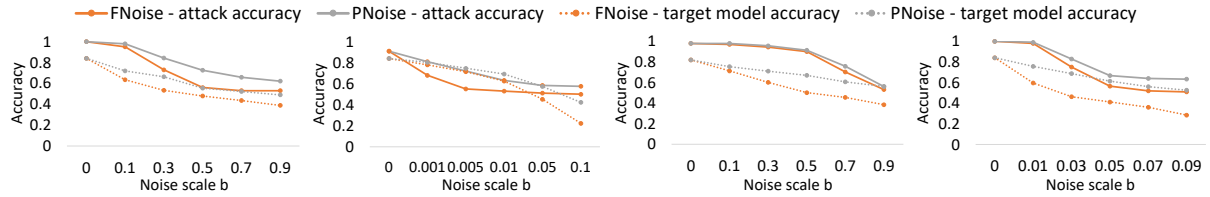
Graph density. Graph density measures the proportion of connected node pairs in the graph. It is formally calculated as $den = \frac{|E|}{|V|\cdot|V|/2}$, where $|V|$ and $|E|$ are the number of nodes and edges, respectively. To assess the impact of graph density on the attack performance, we scale up/down the density ($den = den \times \tau$) of the original graph by inserting or deleting edges while keeping $|V|$ and graph homophily unchanged. We consider density scale values of τ = 0.2, 0.5, 2, 5, and 10.

Table 10 presents the attack performance results for different density scale values. In most settings, the attack performance remains insensitive to changes in graph density. Specifically, for the GRACE, GCA, and CCA-SSG models, the attack accuracy remains stable regardless of the graph density. However, the attack accuracy against the MVGRL model drops dramatically from 0.91 to around 0.5 when the graph density increases, starting from $\tau = 2$. To understand the reason behind this performance drop, we visualize the distribution of member and non-member node pairs for the GRACE and MVGRL models with varying graph densities, and observe that for the GRACE model, member and non-member node pairs remain well distinguishable by their embedding similarity in graphs with different densities. However, for the MVGRL model, while member and non-member node pairs are distinguishable for small densities, they become indistinguishable for dense graphs ($\tau \geq 2$). The reason behind this is that the performance of MVGRL approaches that of random guessing for these dense graphs. This suggests that the embeddings generated by MVGRL do not accurately encode the graph structure information and are of poor quality. Consequently, the attack cannot perform well on these embeddings.

Graph type. Previous studies have indicated that edge augmentation of GCL models has a greater impact on social network graphs compared to other types of graphs, such as biochemical graphs [72]. Therefore, we investigate whether graph types can influence the attack performance. Intuitively, the level of graph homophily depends on the type of the graph. Social network graphs may exhibit higher graph homophily than biochemical graphs. To study the impact of graph homophily on attack accuracy, we consider six graph datasets that belong to three different types with varying

Settings	Cora				Citeseer				Facebook			
	Ours	BL-A	BL-B	BL-C	Ours	BL-A	BL-B	BL-C	Ours	BL-A	BL-B	BL-C
GRACE	0.332	0.181	0.258	0.007	0.519	0.313	0.354	0.009	0.545	0.189	0.293	0.01
MVGRL	0.083	0.026	0.075	0.003	0.084	0.071	0.081	0.003	0.067	0.022	0.054	0.002
GCA	0.124	0.051	0.083	0.022	0.139	0.07	0.111	0.036	0.237	0.096	0.153	0.034
CCA-SSG	0.242	0.127	0.213	0.074	0.318	0.134	0.277	0.089	0.31	0.101	0.19	0.067

Table 5: TPR@0.1%FPR results of GCL-Leak and the three baselines. The best TPR@0.1% FPR performance for each GCL model and each dataset is highlighted with olive color. (BL-A, BL-B, and BL-C represent Baseline-A, Baseline-B, and Baseline-C respectively.)



(a) GRACE (b) MVGRL (c) GCA (d) CCA-SSG

Figure 15: Performance of FNoise and PNoise defense (Cora dataset, $r = 0.4$ for PNoise). $b = 0$ indicates no defense.

GCL model	Cora	Citeseer	Facebook
GRACE	1	0.99	1
MVGRL	0.92	0.9	0.84
GCA	0.98	0.98	0.9
CCA-SSG	1	1	0.98

Table 6: AUC performance of GCL-Leak.

Settings			GRACE	MVGRL	GCA	CCA-SSG
C	D	E				
✓			0.93	0.74	0.59	0.96
	✓		0.73	0.82	0.79	0.95
		✓	0.58	0.61	0.58	0.57
✓	✓		1	0.91	0.97	1
	✓	✓	0.99	0.91	0.96	0.99
✓		✓	0.99	0.89	0.97	0.97
✓	✓	✓	1	0.91	0.97	1

Table 7: Attack accuracy for different combinations of similarity metrics. "C", "D", "E" represent Cosine similarity, Dot Product and Euclidean distance respectively. (Aggregation method: concatenation, Cora dataset).

Settings			Concate-nation	Average	Min pooling	Max pooling
C	D	E				
✓	✓		1	0.76	0.73	0.58
	✓	✓	0.99	0.73	0.73	0.72
✓		✓	0.99	0.76	0.74	0.62
✓	✓	✓	1	0.78	0.74	0.71

Table 8: Attack accuracy for different similarity aggregation methods. "C", "D", "E" represent Cosine similarity, Dot Product and Euclidean distance respectively. (GRACE model, Cora dataset).

levels of graph homophily: (1) *Social network graphs*: We use the Facebook Ego graph[7] and the Google+ graph [27], where nodes represent individual users and edges represent social connections; (2) *E-commerce graphs*: We consider the Amazon Computer [33] and Amazon Photo [33] datasets, where nodes represent products and edges indicate frequent co-purchases; and (3) *Biomedical molecule graphs*: We use the ENZYMES[8] and COX2[9] graphs, where nodes represent atoms and edges represent chemical bonds between atoms.

GCL model	Cora		Citeseer		Facebook	
	w/o F	with F	w/o F	with F	w/o F	with F
GRACE	1	1	1	1	1	1
MVGRL	0.91	0.91	0.9	0.9	0.82	0.83
GCA	0.97	0.97	0.97	0.98	0.91	0.91
CCA-SSG	1	1	1	1	0.95	0.97

Table 9: Attack accuracy of four GCL models with and without feature augmentations. F denotes feature augmentation.

Density scale	0.2	0.5	1	2	5	10
GRACE	1	1	1	1	1	1
MVGRL	0.93	0.91	0.91	0.56	0.53	0.52
GCA	1	0.98	0.97	0.98	1	0.99
CCA-SSG	1	0.9	0.99	0.96	0.94	0.82

Table 10: Impact of graph density on performance of GCL-Leak (Cora dataset).

Table 11 presents the results of attack performance on these seven graphs, ordered by their modularity in ascending order. We observe that the attack accuracy increases with increasing modularity. Particularly, for graphs with high modularity (e.g., modularity ≥ 0.3), the attack accuracy is close to 1. However, when the modularity drops to 0.05 and below, the attack accuracy approaches 0.5 (random guess). We further examine if these observations are

[7] The Facebook Ego graph is a social network graph sampled from the Facebook dataset [27].
[8] We use the pre-processed ENZYMES dataset provided by [17].
[9] We use the pre-processed COX2 dataset provided by [17].

Dataset	Type	Homophily	GRACE	MVGRL	GCA	CCA-SSG
Amazon Computer	E-commerce	0.04	1	0.54	0.97	1
Amazon Photo	E-commerce	0.05	1	0.55	1	1
Facebook ego	Social network	0.1	1	0.62	0.98	0.72
Google+	Social network	0.22	1	0.64	0.99	0.99
ENZYMES	Biochemical	0.3	0.99	0.87	1	0.98
COX2	Biochemical	0.32	0.99	0.84	0.92	0.96

Table 11: Impact of graph type on attack performance of GCL-Leak.

consistent with the previous results on the Cora, Citeseer, and Facebook datasets (Table 5). It turns out that these three datasets also exhibit high homophily values (0.29 for Cora, 0.36 for Citeseer, and 0.09 for Facebook).

H DETAILS OF TWO ALTERNATIVE LUPI APPROACHES

Secondary feature: We adopt the LUPI approach in [61] to our setting to represent the privileged information as regularization terms. Specifically, this approach consists of two classifiers, namely the *primary classifier* that aims to discern edge membership using the similarity-based input features, and the *secondary classifier* that intends to identify edge membership through the use of the secondary feature.

The augmented membership of each node pair is treated as a secondary feature to the secondary classifier. Specifically, the secondary feature $\tilde{x}_{j,k}$ of the node pair (v_j, v_k) is designed as follows:

$$\tilde{x}_{j,k} = ||_{\forall Aug^i(v_j,v_k)} Aug^i(v_j, v_k), \tag{10}$$

where $Aug^i(v_j, v_k) = y^i_{j,k}||...||y^i_{j,k}$, with $y^i_{j,k}$ the membership label of the node pair (v_j, v_k) in the augmented view G^i. The length of $Aug^i(v_j, v_k)$ is the same as the number of similarity metrics, i.e., the same as the length of $Csim^i(v_j, v_k)$, $Csim^i(v_j, v_k)$ follows Eqn. (3).

Based on both classifiers, the loss of the secondary classifier is added to the primary classifier as the penalty. The objective function of the primary classifier is formulated as follows:

$$\underset{w,\tilde{w}}{argmin} \sum_{(v_j,v_k\in G^S)} \mathcal{L}_{Pri}(y_{j,k}, x_{j,k}, w) + \eta \mathcal{L}_{Sec}(y_{j,k}, \tilde{x}_{j,k}, \tilde{w}), \tag{11}$$

where w and $\tilde{w}$ denote the parameters of the primary classifier and the secondary classifier respectively, $x_{j,k}$ and $\tilde{x}_{j,k}$ denote the primary and secondary features of the primary classifier and the secondary classifier respectively, where $x_{j,k}$ follows either Eqn. (4) or Eqn. (5), depending on whether $x_{j,k}$ is a member or a non-member, and $\tilde{x}_{j,k}$ follows Eqn. (10). Furthermore, $y_{j,k}$ and $y^i_{j,k}$ in Eqn. (11) denote the ground-truth membership label and augmented membership label of (v_j, v_k), $\mathcal{L}_{Pri}$ and $\mathcal{L}_{Sec}$ denote the loss function of the primary and secondary classifier, and η denotes a scaling parameter. A higher value of η indicates a greater penalty imposed by the secondary classifier. In this paper, we use Binary-Cross-Entropy (BCE) loss for both $\mathcal{L}_{Pri}$ and $\mathcal{L}_{Sec}$. By introducing the loss of the secondary classifier as the regularizer term to the loss function of the primary classifier, the "identical-membership" information is incorporated with the attack classifier.

Secondary target: We employ the LUPI approach in [6] to represent the privileged information as a secondary target. Specifically, we view the learning of the attack classifier as a multi-learning task, where one of the tasks is to discriminate the membership of node pairs based on ground truth memberships, and the other task is to discriminate the membership of node pairs based on memberships in various augmentations. Thus the objective function of the attack classifier is formulated as follows:

$$\begin{aligned}\underset{w}{argmin} \quad & \lambda \sum_{(v_j,v_k\in G^S)} \sum_{i=0}^{N} \mathcal{L}(y^i_{j,k}, Csim^i(v_j, v_k), w) \\ & + (1-\lambda) \sum_{(v_j,v_k\in G^S)} \sum_{i=0}^{N} \mathcal{L}(\tilde{y}^i_{j,k}, Csim^i(v_j, v_k), w),\end{aligned} \tag{12}$$

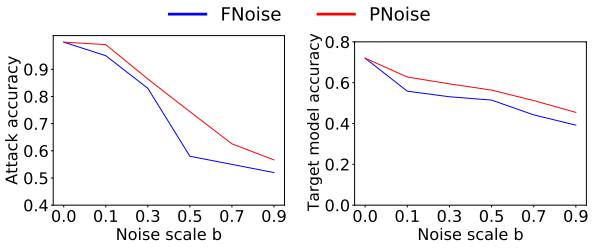
where $Csim^i(v_j, v_k)$ follows Eqn. (3), $y^i_{j,k}$ and $\tilde{y}^i_{j,k}$ denote the ground truth membership label and augmented membership label of (v_j, v_k) respectively, $\mathcal{L}$ denotes the loss function of the attack classifier, M denotes the total number of the augmented views, and λ is a scaling parameter. In this paper, we use Binary-Cross-Entropy (BCE) loss for $\mathcal{L}$. The learning objective is to learn the parameters w of the attack classifier, with $\tilde{y}^i_{j,k}$ as the secondary label for the training instances, with the loss from the secondary label $\tilde{y}^i_{j,k}$ as a penalty term of the attack classifier. A higher λ value indicates a greater penalty imposed by the secondary target.

As the classifier will predict a label for each augmented view, we employ majority voting of the labels of all the augmented views to determine the membership of (v_j, v_k) in the training graph.

I ADDITIONAL RESULTS ON DEFENSE MECHANISMS

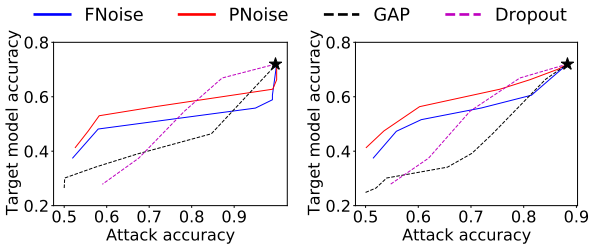
I.1 Defense performance for Four GCL Models

Figure 15 showcases the performance of FNoise and PNoise across various noise levels on four GCL models. Regarding FNoise, we have the following observations. First, FNoise proves to be effective, reducing the attack accuracy to approximately 0.5 for all four GCL models. Second, while all models exhibit the same trend of lower attack accuracy with higher noise scale, their sensitivity to the defense varies. For instance, MVGRL's attack accuracy can be reduced to 0.5 with a noise scale of $b = 0.005$, while CCA-SSG requires $b = 0.05$, and GRACE necessitates $b = 0.5$. GCA is the least sensitive to the defense mechanism, requiring a noise scale as large as 0.9 to achieve an attack accuracy of around 0.5. Third, there exists a trade-off between privacy and GCL model accuracy. A stronger defense comes with the lower target model accuracy. However, the target model accuracy remains significantly higher than random guess (which is $\frac{1}{6} \approx 0.17$ for the Cora dataset) even when the attack



(a) Attack performance (b) Target model accuracy

Figure 16: Performance of FNoise and PNoise (GRACE model, Citeseer dataset, $r = 0.4$ for PNoise). $b = 0$ indicates no defense.



(a) GCL-Leak as attack (b) W-LMIA [8] as attack

Figure 17: Target-attack trade-off of the defense mechanisms against GCL-Leak and W-LMIA [8] (GRACE model, Citeseer dataset, $r = 0.4$ for PNoise). ⋆ indicates no defense.

accuracy is reduced to around 0.5. For instance, for FNoise when the attack accuracy is approximately 0.5 ($b = 0.005$, Figure 15 (b)), the target model accuracy for MVGRL is 0.73. Similarly, when the attack accuracy is around 0.5 ($b = 0.05$, Figure 15 (d)), the target model accuracy for CCA-SSG is 0.41. This indicates that FNoise effectively addresses the trade-off between defense and target model accuracy. As for PNoise, the observations align closely with FNoise. However, FNoise demonstrates a stronger defense capability than PNoise, as PNoise requires greater noise to achieve an attack accuracy near 0.5 than FNoise. On the contrary, PNoise offers a more favorable trade-off compared to FNoise due to its approach of selectively introducing noise into embedding dimensions of lesser importance.

I.2 Defense Performance for Citeseer Dataset

Figure 16 illustrates the defense performance of FNoise and PNoise on the Citeseer dataset. We make the following observations: First, both FNoise and PNoise defense mechanisms successfully reduce the attack accuracy to approximately 0.5. This confirms the effectiveness of the proposed defense mechanism. Second, although FNoise and PNoise exhibit the same trend of lower attack accuracy with a higher noise scale, their sensitivity to the defense varies that FNoise is more effective than PNoise with the same noise scale b. These variations in sensitivity are also observed in Figure 11 (a). Third, a trade-off exists between privacy and embedding quality on both FNoise and PNoise defense mechanisms, however, the PNoise method exhibits a lower target model loss compared to the FNoise method when the attack accuracy is close to 0.5. This difference can be attributed to the PNoise approach selectively introducing noise to the embedding dimensions of lesser importance. Consequently, this results in a reduced loss when compared to the FNoise method.

I.3 Defense-utility Trade-off on Citeseer Dataset

Figure 17 illustrates the defense-utility trade-off of our two defense methods and the two baseline approaches on the Citeseer dataset. The observations are similar to Figure 12 and thus we omit the discussions.

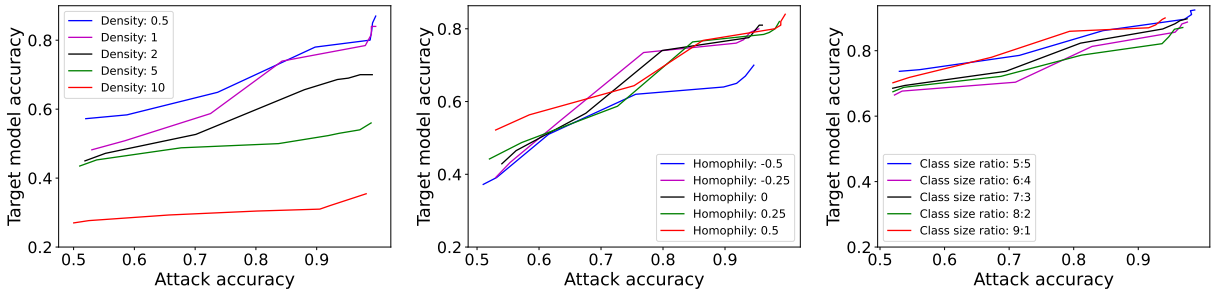
I.4 Factor Analysis of Defense-utility Trade-off

We investigate how various factors such as graph density, graph homophily, and class distribution impact the defense-utility trade-off performance of our defense method. We only consider PNoise in this set of experiments as it has a better defense-utility trade-off than FNoise (Figure 11 and Figure 16)

Impact of graph density. Graph density has been shown as one of the factors that affect LMIA accuracy [83]. Thus, we investigate whether it can impact on defense-utility trade-off too. We change the graph density $d' = d \times \tau$ by inserting/deleting edges while keeping the graph homophily unchanged, where τ is the scale factor. We use $\tau = 0.2, 0.5, 2, 5$ in the experiments, and illustrate the defense-utility ROC curve for each τ value in Figure 18 (a). Notably, graphs with lower density exhibit higher target model accuracy when subjected to the same level of attack accuracy. In other words, our defense method achieves better defense-utility trade-off on sparse graphs than dense ones.

Impact of graph homophily. We vary the network modularity (Eqn. 9) by adding/deleting edges. Figure 18 (b) illustrates the defense-utility ROC curve for five different network modularity values. It shows that graph homophily has a minimal impact on the defense-utility trade-off. Specifically, our defense method demonstrates slightly improved trade-off on graphs with higher homophily compared to those with lower homophily, particularly when the attack accuracy is below 0.6.

Impact of class distributions. We consider the dataset that includes only two classes, and change the distributions of these two classes by randomly flipping the labels. Figure 18 (c) illustrates the defense-utility ROC curve for five distinct class distributions, with size ratios between the two classes set at 5:5, 6:4, 7:3, 8:2, and 9:1. We do not observe any discernible impact of class distribution on the trade-off between attack accuracy and model accuracy.



(a) Impact of graph density **(b) Impact of graph homophily** **(c) Impact of class distribution**

Figure 18: Defense-utility Trade-off performance of our defense (Cora dataset)