GRID: Protecting Training Graph from Link Stealing Attacks on GNN Models

Jiadong Lou*, Xu Yuan*, Rui Zhang*, Xingliang Yuan†, Neil Zhenqiang Gong‡, and Nian-Feng Tzeng§
*University of Delaware, †The University of Melbourne,
‡Duke University, §University of Louisiana at Lafayette

***Abstract*—Graph neural networks (GNNs) have exhibited superior performance in various classification tasks on graph-structured data. However, they encounter the potential vulnerability from the link stealing attacks, which can infer the presence of a link between two nodes via measuring the similarity of its incident nodes' prediction vectors produced by a GNN model. Such attacks pose severe security and privacy threats to the training graph used in GNN models. In this work, we propose a novel solution, called Graph Link Disguise (GRID), to defend against link stealing attacks with the formal guarantee of GNN model utility for retaining prediction accuracy. The key idea of GRID is to add carefully crafted noises to the nodes' prediction vectors for disguising adjacent nodes as n-hop indirect neighboring nodes. We take into account the graph topology and select only a subset of nodes (called *core nodes*) covering all links for adding noises, which can avert the noises offset and have the further advantages of reducing both the distortion loss and the computation cost. Our crafted noises can ensure 1) the noisy prediction vectors of any two adjacent nodes have their similarity level like that of two non-adjacent nodes and 2) the model prediction is unchanged to ensure zero utility loss. Extensive experiments on five datasets are conducted to show the effectiveness of our proposed GRID solution against different representative link-stealing attacks under transductive settings and inductive settings respectively, as well as two influence-based attacks. Meanwhile, it achieves a much better privacy-utility trade-off than existing methods when extended to GNNs.**

1. Introduction

Graph-structured data, where nodes represent entities and edges describe their relationships, are prevalent in various real-world applications, such as social networks, molecular graphs, and natural language processing. To handle such data, Graph Neural Networks (GNNs) [30], [10], [38] were developed to effectively learn node features and capture the complex interactions among nodes, enabling better performance in tasks like node classification and link prediction. Despite the rapid advancement of GNN models, significant security and privacy concerns have surfaced, particularly regarding sensitive information leakage within the graph topology, such as link interactions. Accordingly, link stealing attacks were introduced in [11], [45], [21], [46], [50], aiming to extract the topology information of a training graph by inferring the presence of a link between any two queried nodes. The intuition behind link inference attacks stems from the fact that the message-passing mechanism of GNNs aggregates information from neighboring nodes for each node, resulting in distinctive patterns in the predictions of neighboring nodes, such as high similarity. Such attacks, which extract links in the training graph of target GNN models, can lead to user privacy leakage, intellectual property infringement, among others.

Witnessing such a link leakage risk, it is unfortunate that existing defensive solutions designed to thwart machine learning inference attacks are unsuitable for defending against link-stealing attacks. These defensive methods can be broadly grouped into three categories: 1) by adding additional terms in loss functions to obscure training dataset patterns [32], [23]; 2) by adding crafted perturbations to the training process via differential privacy techniques [31], [4], [40], [48], [45], [26]; 3) by adding noises to the prediction vectors of the trained model [15]. The first two categories of solutions will interfere with the model training process, inevitably hurting model performance to some extent. For example, the pioneering differential privacy method as reported in [26] incurs over 20% model accuracy degradation when defending against the node inference attacks. The third category of solution aims to craft the noise independently for each prediction vector, thereby failing to consider the similarity patterns among adjacent nodes governed by the GNN model and the graph structure. Hence, it is urgent to call for a new defensive scheme that can take into account both the graph topology and GNN information aggregation characteristics to protect the GNN models from link-stealing attacks while formally ensuring model utility guarantees.

This paper presents a novel defense method, named **Gr**aph **L**ink **D**isguise (**GRID**), to prevent link-stealing attacks against GNN models, being the first to provide a formal guarantee for model utility. GRID adds noise vectors to the prediction vectors of nodes in the training graph to distort the similarity of adjacent nodes, thereby obscuring the existence of links. We formulate this defense scheme as an optimization problem to achieve two goals: 1) preventing the attacker's link-stealing classifiers from determining if a link exists between two nodes based on their queried prediction vectors, and 2) maintaining the accuracy of the model's prediction results. However, a number of challenges arise in solving the noise generation problem. First, the inherent data aggregation among neighboring nodes in GNN models makes it difficult to dissociate the correlation between similar predictions and link presence, while only slight noises

© 2025, Jiadong Lou. Under license to IEEE.
DOI 10.1109/SP61157.2025.00059

are tolerant for retaining model predictions. Second, given the large and complex real-world graph topologies, simply adding noise to the prediction vectors of all nodes results in unacceptable computational complexity and noise offset. Therefore, to avert conflicts and improve computational efficiency, graph topology information should be considered when crafting the noise vectors, which represents another huge challenge.

To tackle the aforementioned challenges, we propose a novel solution with the key idea of disguising adjacent nodes as n-hop indirect neighboring nodes and crafting noise only for selected core nodes' prediction vectors. First, instead of minimizing the prediction similarity of adjacent nodes, we disguise them by reducing their similarity patterns to the level of indirectly connected neighboring nodes. This approach is sufficient to disrupt link inference and thwart adaptive attacks, as adjacent nodes' prediction vectors will no longer exhibit distinctive similarity patterns. Additionally, considering the interconnections among nodes, we identify a subset of nodes (dubbed core nodes) that cover all links in the training graph. By targeting noise additions only to core nodes, we can avoid conflicts among nodes and reduce computational complexity. Thus, we reformulate our defense problem by minimizing the similarity gap between adjacent nodes and indirectly connected neighboring nodes, calculating the necessary noise for core nodes accordingly.

We conduct extensive experiments on five graph datasets (comprising three citation datasets and two chemical datasets) and different GNNs (both transductive and inductive models) under representative link-stealing attacks proposed in [11], [46] to evaluate the effectiveness of our GRID method. Our experimental results demonstrate that GRID can substantially degrade the inference performance of all these link-stealing attacks, reducing their accuracy to around 50% (similar to random guessing) with appropriate parameter settings. Additionally, we compare our GRID with five state-of-the-art defense approaches across three categories: employing regularization terms, applying differential privacy techniques, and adding prediction noise [32], [36], [23], [15], [26]. The comparative results show that our GRID significantly outperforms all five counterparts in terms of the defense-utility trade-off.

Our contributions can be summarized as follows

- We propose the first Graph Link Disguise method, i.e., GRID, to prevent link stealing attacks on GNN models with the formal utility guarantee, serving as a new framework for protecting the graph topology information.
- We propose a novel scheme of crafting noises for a subset of nodes to disguise the adjacent nodes into n-hop indirect neighboring nodes, able to defend against link inference attacks.
- We conduct comprehensive experiments to demonstrate that our GRID effectively defends against link-stealing attacks and outperforms all existing methods in terms of the defense-utility trade-off.

2. Background

2.1. Graph Neural Networks

Graph neural networks (GNNs) are neural network techniques tailored for learning from graph-structured data. Typically, a node classification GNN model, denoted as $f_g()$, takes the graph topology with a set of labeled nodes and their attributes as its input for training. This model is then used to predict the labels of remaining unlabeled nodes. That is, the training dataset can be expressed as $\mathcal{G} = (\mathcal{N}, \mathcal{A}^{\mathcal{N}}, \mathcal{F}, \mathcal{L})$, where $\mathcal{N}$ denotes a set of nodes, $\mathcal{A}^{\mathcal{N}}$ represents the adjacency matrix which depicts the graph topology, $\mathcal{F}$ represents node attributes, and $\mathcal{L}$ indicates labels. The node classification model $f_g()$ takes $\mathcal{G}$ for training and then predicts the labels of unlabeled nodes.

GNNs can be categorized as transductive and inductive learning paradigms. In transductive learning, GNNs are trained on both labeled and unlabeled nodes of a fixed graph and then used to predict the labels of the unlabeled nodes within this graph. It is typically used in scenarios where the graph structure is fixed, such as predicting protein functions or gene associations within the detected protein-protein interaction or gene regulatory graphs, with Graph Convolutional Networks (GCNs) being the classic model proposed for this paradigm. In inductive learning, GNNs are trained on labeled nodes and can predict the labels of nodes unseen during the training process, with Graph Attention Networks (GATs) [38], Graph Transformer Networks (GTNs) [49], among others being proposed. Inductive GNNs are used in dynamic scenarios where new nodes or edges can appear, and the model needs to generalize to unseen data, such as providing recommendations for new users in recommendation systems and analyzing new users and their potential connections in a dynamic social network.

2.2. Link Stealing Attacks on GNNs

The link inference of GNN models was first proposed in [11], called the link stealing attacks, which aim to infer the presence of a link between two nodes in the training graph. In these attacks, an attacker can query data samples over a trained GNN model, to obtain the corresponding prediction vectors. Based on this, the attacker attempts to infer whether a link exists between the queried nodes n_i and n_j in the training graph. The intuition of different kinds of link inference attacks stems from the same fact that GNNs aggregate information for each node from its neighboring nodes. According to this inherent GNN nature, two lines of attacks are proposed.

2.2.1. Similarity-based Attack. In [11], authors first propose to exploit the characteristic in the prediction vectors of two neighboring nodes that if two nodes are linked, their prediction vectors queried from the target model should be similar, as they have shared some similar information. Therefore, the attacker can learn the similarity difference between the prediction vectors of linked or unlinked node

TABLE 1. EIGHT ATTACKS AND THEIR BACKGROUND KNOWLEDGE

Attack	$\mathcal{F}$	$\mathcal{A}^*$	$\mathcal{D}'$	Attack	$\mathcal{F}$	$\mathcal{A}^*$	$\mathcal{D}'$
Attack-0	×	×	×	Attack-4	×	✓	✓
Attack-1	×	×	✓	Attack-5	✓	×	✓
Attack-2	✓	×	×	Attack-6	✓	✓	×
Attack-3	×	✓	×	Attack-7	✓	✓	✓

pairs to infer whether there is a link between them. Mathematically, given two nodes i and j, the attacker aims at training a link-stealing classifier, denoted as $f_m()$, to have

$$f_m\big(\mathcal{E}(\mathbf{v_i}, \mathbf{v_j})\big) = \begin{cases} 1 & \text{if } \mathcal{A}_{ij} = 1 \ ; \\ 0 & \text{otherwise} \ , \end{cases} \tag{1}$$

where $\mathbf{v_i} = f_g(n_i)$, $\mathbf{v_j} = f_g(n_j)$, representing their prediction vectors, and $\mathcal{E}()$ captures the similarity metrics. $\mathcal{A}_{ij} = 1$ represents there is a link between n_i and n_j. In practice, the link-stealing classifier's output, i.e., $f_m()$, is a confidence score in the range of 0 to 1. If it is greater than 0.5, the classifier is confident to predict that there is a link between the two queried nodes; otherwise, no link exists.

In [11], eight types of attacks (defined as Attack-0 to Attack-7) were proposed as shown in Table 1, assuming an attacker has different levels of background knowledge, including 1) partial node attributes from the target dataset, 2) partial graph topology of the target dataset and 3) a shadow dataset. Among these, Attack-0 and Attack-1 are two representative attacks that cover unsupervised and supervised black-box scenarios, while Attack-6 is the strongest gray-box attack, where the attacker can access partial graph topology and node attributes. Although initially designed for transductive models, these attack schemes have also been shown to be effective on inductive GNN models, as both models output predictions for nodes in the training graph regardless of whether they are applied in transductive or inductive tasks. For example, attacks proposed in [46], abbreviated as ILS, extend these ideas to inductive GNN models by calculating and differentiating metrics between the prediction vectors of pairwise nodes. Ten attacks were proposed to cover various scenarios, with Attack A2 and A9 being reported as the strongest in the posterior-only and combined attack categories, respectively. These similarity-based link-stealing attacks are both general and practical, posing a significant threat to the training graphs.

2.2.2. Influence-based Attack. The second type of link-stealing attack was proposed in [45], [21], which leverages the observation that two nodes are more likely to be linked if changing the features of one node significantly influences the prediction results of the other node due to the message-passing mechanism. To effectively capture such influence delivered through the link, the authors introduce the graph data poisoning scenario where an attacker can perform malicious node injection (also called infiltration) into the training graph or maliciously change the attributes of the controlled nodes during the inference stage.

Taking the latest attacks proposed in [21] as an example, the attacker aims at inferring the link between two target nodes u and v. The attacker first injects a node a and a link (a, u) to the training graph $\mathcal{G}$. Then it queries the target model to obtain the prediction for node a, denoted as $f_g(a)$. Second, the attacker injects another node b and a link (b, v) to the training graph $\mathcal{G}$ and again queries the target model to obtain the prediction for node a, denoted as $f_g(a|b)$. If u and v are linked, the two predictions $f_g(a|b)$ and $f_g(a)$ should have a larger difference as the features of node b will influence the prediction of node a through the link (u, v). Formally, if $||f_g(a|b) - f_g(a)||_2 > \eta$, where η is an empirical threshold, the attacker determines that there is a link between u and v.

While influence-based attacks have shown better performance than similarity-based attacks in certain scenarios [45], [21], their practical applicability remains limited in certain contexts. First, these attacks are not robust to graph dynamics, as the influence is primarily determined by a fixed threshold rather than the attributes of the injected node. For instance, in social network scenarios, consider two users, u and v, who are not linked. The attacker follows the method in [21], linking malicious users a and b to target users u and v. The attack observes the influence of b on a by calculating the difference between the predictions $f_g(a|b)$ and $f_g(a)$. However, if a normal user links to u (follows u), a common occurrence in real-world applications, it can cause a significant difference between $f_g(a|b)$ and $f_g(a)$. This misleads the attacker into identifying the change as the influence of b on a through the non-existent link between u and v, resulting in substantial false predictions under practical conditions. Second, these attacks rely on querying the GNN model with newly injected nodes that have not been part of the training process, rendering them unsuitable for transductive models. Given these considerations, our work focuses primarily on defending GNN models from similarity-based link-stealing attacks. However, we will also evaluate the effectiveness of our proposed defense against influence-based attacks in Section 6.1.

3. Problem Statement

This section clarifies our threat model and defense objectives and then formulates our defense problem.

3.1. Threat model

Target Model. We focus on the GNN model $f_g()$ that has been trained on a graph dataset $\mathcal{G} = (\mathcal{N}, \mathcal{A}^{\mathcal{N}}, \mathcal{F}, \mathcal{L})$, for node classification tasks. Given a node n_i as a querying sample, the trained GNN model will output a prediction vector v_i, representing the probability distribution that the querying node belongs to each class.

Attacker. The attacker's goal is to perform similarity-based link-stealing attacks, attempting to determine whether there exists a link between two nodes. The attacker has black-box access to the target GNN model $f_g()$. He can query the model with nodes to obtain their prediction vectors, which are then leveraged to infer whether the two nodes are linked or not. We follow the same setting as in [11], by considering three types of background knowledge that an attacker may

possess. In addition, we consider the challenging scenario in which the attacker can know the defense mechanism adopted by the model provider to adapt his attack method.

Model Provider. The model provider trains the target GNN model $f_g()$ and makes it available for public or commercial query access. Under our threat model, the model provider's goal is to defend against potential link-stealing attacks in order to protect its training graph. Naturally, the model provider has complete knowledge of the target model $f_g()$ and its training graph. On the other hand, regarding different link-stealing attacks, the model provider may not know exactly which type of attack is to be launched. Hence, to mitigate all potential threats, he has to develop a mechanism capable of defending against all types of attacks.

3.2. Defense Formulation

In this paper, we aim to propose a defense method for the model provider to protect its GNN model from link-stealing attacks. Our goal is twofold: 1) The defense method should prevent the potential link-stealing attack from inferring the presence or absence of a link between two querying nodes and 2) the target model's prediction utility should not be compromised. Next, we detail our motivation and provide mathematical formulations for our defense methods.

3.2.1. Our Motivations. Although multiple attack schemes differ in implementation details to encompass various attack scenarios, they all leverage the same pattern: the similarity of prediction vectors from adjacent nodes is greater than that of non-adjacent nodes. As such, our defense method should aim to distort this similarity. By doing so, attacks that rely on distinguishing the similarity of an adjacent node pair from that of a non-adjacent node pair will become ineffective.

However, this similarity pattern arises from the data aggregation from neighboring nodes, which constitutes the fundamental design integral to the GNN model's efficacy. Dissociating the correlation between prediction similarity and the attributes of node adjacency during the model training process will inevitably degrade model performance. Therefore, we propose adding noise to the prediction vectors for the nodes in the training graph after the model has been trained to disguise the high similarity between adjacent nodes' prediction vectors. That is, the model provider can first train the GNN model and then append the noise to the prediction vector output from the trained GNN model. Hence, the attacker can only obtain the noisy predictions for the training nodes, which makes it hard to steal the link. Formally, the noisy prediction vector $\mathbf{r}$ is expressed as

$$\mathbf{r} = \mathbf{v} + \mathbf{s} \ , \tag{2}$$

where $\mathbf{s}$ represents the noise vector calculated by our method corresponding to a prediction vector $\mathbf{v}$ for a training node from the target GNN model. Additionally, we will formulate this noise generation as an optimization problem to ensure that both defense goals and model utility are maintained.

3.2.2. Defense Goal. To defend against the similarity-based link-stealing attack, the noise calculated for the prediction vectors of two neighboring nodes must reduce their high similarity. For any two neighboring nodes i and j, we quantify the similarity of their prediction vectors $\mathbf{v}_i$ and $\mathbf{v}_j$, denoted as $sim(\mathbf{v}_i, \mathbf{v}_j)$, as follows

$$sim(\mathbf{v}_i, \mathbf{v}_j) = corr(\mathbf{v}_i, \mathbf{v}_j) + cos(\mathbf{v}_i, \mathbf{v}_j) \ , \tag{3}$$

where $corr()$ and $cos()$ measure the Pearson correlation coefficient and cosine metrics, with detailed calculations shown in Appendix A.1. We choose correlation and cosine metrics since they were observed in [11] to better facilitate pattern extraction for link-stealing attacks. Thus, our defense objective can be formulated as an optimization problem of noise calculation for similarity minimizing, yielding:

$$\text{OPT-1: } \min \sum_{n_i.n_j \in \mathcal{N}, \mathcal{A}_{ij}=1} sim(\mathbf{v}_i, \mathbf{v}_j) \ . \tag{4}$$

3.2.3. Utility Goal. To ensure that the target model's prediction performance is not compromised, we introduce three constraints that consider the model's prediction utility from different perspectives, as detailed below.

Prediction Loss. We first consider the prediction loss, which measures the label predicted by a GNN model for the queried node. Preventing prediction loss is critical since, in some applications such as healthcare, even 1% of wrongly predicted labels are intolerable and may incur irreparable consequences. Hence, to maintain the target model's prediction performance, we aim to achieve zero prediction loss in our defense mechanism, i.e., our added noises should not alter the predicted label for any queried node. Recall that the predicted label class of a node is determined by the largest confidence score in its prediction vector. Such a prediction loss constraint can be expressed by

$$\arg\max_a \{v^a\} = \arg\max_a \{v^a + s^a\} \ , \tag{5}$$

where v^a and s^a respectively represent the a-th element in the prediction vector $\mathbf{v}$ and in the noise vector $\mathbf{s}$. Eqn. (5) constrains that the largest element in a prediction vector will remain the same before and after adding the noises, thus keeping the label unchanged.

Probability Distribution. The second utility constraint is that the prediction vector should remain a probability distribution, with each element representing the probability of being classified into a particular class, and the sum of all elements is equal to 1. The newly added noise vectors should follow this property, yielding:

$$0 \leq v^a + s^a \leq 1, \sum_a (v^a + s^a) = 1, \forall v^a \in \mathbf{v}, s^a \in \mathbf{s} \ . \tag{6}$$

Distortion Budget. The third utility constraint aims to preserve latent information in the prediction vectors, which are deemed important when applying them as feature vectors for other downstream tasks. Hence, we need to minimize the difference between the original and altered prediction vectors. Specifically, the model provider can specify an

allowable distortion budget θ, representing the maximum alteration to any prediction vector for constraining altered latent information. Mathematically, we have

$$d(\mathbf{v}, \mathbf{v}+\mathbf{s}) \leq \theta \ , \tag{7}$$

where $d()$ is the distance calculation to measure the difference between the original and distorted prediction vectors, with the L-1 norm adopted for such distance calculation.

3.3. Defense Formulation

Based on our goals, we can mathematically formulate our defense scheme as an optimization problem, i.e.,

$$\text{OPT-2: } \min \sum_{n_i, n_j \in \mathcal{N}, \mathcal{A}_{ij}=1} sim(\mathbf{v}_i, \mathbf{v}_j)$$
$$s.t. \ (3), (5), (6), \text{and } (7),$$

where prediction vectors v in the formulation are constant and are produced by the trained GNN model. The noise vectors s are variables to be solved. This formulation aims to find the optimal noise vectors to achieve the defense goal.

4. GRID: Graph Link Disguise

This section delves into the challenges in solving the proposed optimization problem and presents our intuitions for resolving them, i.e., our Graph Link Disguise (GRID).

4.1. Key Challenges

Three challenges arise when solving the optimization problem OPT-2 in practice.

The first challenge arises from the inherent difficulty within GNNs of dissociating the correlation between prediction similarity and the attributes of adjacent nodes. The message passing and data aggregation mechanisms of the GNN model inevitably cause an apparently high degree of similarity between the prediction vectors of adjacent nodes. For example, our experiments on the Cora dataset [42] reveal that the mean cosine similarity is 0.94 for adjacent nodes, compared to 0.63 for non-adjacent nodes. While adding heavy noise vectors to the prediction vectors can reduce this similarity, as formulated by our mathematical constraints, only slight noise can be tolerated to preserve the model's utility and prediction performance. Thus, the significant difference in similarity due to the GNN poses considerable challenges in distorting the similarity among adjacent nodes while maintaining utility.

The second challenge originates from the computational complexity and noise offset arising from the incorporation of noise. In OPT-2, all nodes are considered for noise addition. If we simultaneously optimize the noise vectors for all nodes, the computational complexity and time cost become unacceptable in practice, especially with a large number of nodes in the training graph. On the other hand, if we calculate the noise individually for each pair of neighboring nodes, the independently crafted noise vectors may counteract each other during similarity calculations, rendering them ineffective in preventing an attacker from inferring links. This issue is exacerbated when a node is linked to multiple nodes in the graph. Therefore, when crafting noise vectors, it is crucial to consider the graph topology to address these problems, posing a significant challenge in our design.

The third challenge stems from the need to take into account the potential adaptive attacks. We consider the most challenging scenario, where the attacker has complete knowledge of our defense scheme, thereby being able to conduct a tailored adaptive attack. If we blindly minimize the similarity between adjacent nodes, the attacker can recognize such a strategy and infer the presence of a link when the similarity of the prediction vectors of a querying node pair is below a certain level. Hence, enabling our defense to counter the adaptive attack is challenging.

4.2. Our Intuitions

To tackle the first and third challenges, we propose the Graph Link Disguise solution which focuses on diminishing the prediction similarity values of adjacent nodes to be commensurate with those of n-hop indirect neighboring nodes rather than minimizing them as much as possible. This commensurate way is more attainable as the required reduction in the similarity of adjacent nodes is less, able to effectively cope with the first challenge. Besides, with the prediction similarity of adjacent nodes akin to that of n-hop indirect neighboring nodes rather than becoming lowest, this method can camouflage adjacent nodes as indirectly connected neighboring nodes, able to defend against adaptive attacks as described in the third challenge. Hence, our goal becomes to minimize the gap between the similarity of adjacent nodes and that of n-hop indirect neighboring nodes.

To tackle the second challenge incurred by noise addition on all nodes, we select a subset of nodes to form a core node set that encompasses nodes linked to all graph nodes. This set is known as the vertex cover set in graph theory [43], which includes at least one endpoint node of every edge in the graph. Since all nodes are linked to these core nodes, the similarity of every pair of adjacent nodes can be altered by the noise crafted for the core nodes' prediction vectors. By focusing only on these core nodes for noise crafting, we can substantially reduce computational complexity. Additionally, because only one node in each edge is considered, we can calculate the noise individually for each pair of neighboring nodes while mitigating noise offset.

4.3. GRID Solution

Following our intuitions, we rectify the optimization problem of OPT-2 in calculating the noise vectors.

4.3.1. Link Disguise. Here, we achieve the link disguise by minimizing the gap between the similarity of adjacent nodes and of n-hop indirect neighboring nodes. Considering

a node i, its adjacent node set is denoted as $\mathcal{P}_i$, and its n-hop indirect neighboring nodes are denoted as $\mathcal{Q}_i$. The similarity gap denoted as D_i, under corresponding noise vector $\mathbf{s}_i$, is calculated by

$$D_i = \sum_{j \in \mathcal{P}_i} sim(\mathbf{v}_i + \mathbf{s}_i, \mathbf{v}_j) - \sum_{k \in \mathcal{Q}_i} sim(\mathbf{v}_i + \mathbf{s}_i, \mathbf{v}_k) \ . \quad (8)$$

This formulation captures the gap between the similarity of a node i to its adjacent nodes and the similarity of this node to its n-hop indirect neighboring nodes, achieved by adding the noise vector s_i. As such, the objective function in OPT-2 can be transformed into the following form:

$$\min \sum_{i \in \mathcal{N}} D_i \ . \quad (9)$$

This objective function aims at minimizing the similarity gap corresponding to all nodes.

4.3.2. Threshold-based Core Nodes Selection. We denote the set of core nodes as $\mathcal{N}_c$, which should cover all links in the training graph $\mathcal{G}$. That is, any link in the graph should connect to at least one node in $\mathcal{N}_c$. Intuitively, we should identify the minimum vertex cover set as the core nodes so that the number of noise vectors added by GRID is minimized. However, finding the minimum vertex cover set is an NP-hard problem. Fortunately, our scenario does not require the core node set to be exactly the same as the minimal vertex cover set, due to the following reasons. First, our objective is to degrade the similarity level of adjacent nodes to that of n-hop indirect neighboring nodes. If the similarity of a given linked node pair is already low enough (e.g., lower than the average similarity of n-hop indirect neighboring nodes), it is unnecessary to include its corresponding nodes. Second, a high-degree node is linked with more nodes, resulting in a broad range of similarity values for its related links. Selecting these high-degree nodes would require substantial amounts of noise (referred to as heavy noise) to modify their prediction vectors and alter the similarity values of all connected nodes. Third, from a mathematical perspective, the cosine similarity metric we employ is highly sensitive to the original value, requiring more effort (i.e., heavy noise) to degrade smaller cosine similarity values. Considering these factors, we propose a threshold-based core node selection approach, by setting a threshold δ as the averaged similarity value of n-hop indirect neighboring nodes.

The proposed threshold-based core node selection approach is depicted next, with its pseudo-code listed in Algorithm 1. The input is training graph $\mathcal{G}$ with the node set $\mathcal{N}$, adjacency matrix $\mathcal{A}$, and a preset threshold δ. Its output is a set of core nodes $\mathcal{N}_c$. Specifically, the threshold δ, determined by the averaged similarity value of n-hop indirect neighboring nodes, is approximated using a subset of n-hop indirect node samples from the training graph to reduce computational complexity. Our algorithm consists of three steps. First, we extract the edge list $\mathcal{E}$ from the adjacency matrix $\mathcal{A}$ and calculate the similarity value of the neighboring nodes of each edge as the weight for sorting the edges in descending order. Second, we sum the similarities of each node with all its adjacent nodes as its degree. Third, we iterate through each edge in our sorted list to select the core node set. That is, an edge in this list is dropped, if its similarity metric is less than the threshold δ; otherwise, if both vertices of this edge are not yet in the vertex cover, we add the one with a higher degree to the core node set. This process continues until all edges in the sorted list have been checked. The result is a set of core nodes $\mathcal{N}_c$ which cover all graph edges having weights above the defined threshold. The computation cost for Algorithm 1 is dominated by extracting the edge list and then calculating prediction similarity on each link. So, Algorithm 1 requires examining each node pair when the graph topology is stored in the adjacency matrix, which leads to a time complexity of $O(n^2)$ where n is the number of nodes. The calculation of similarity has a complexity of $O(e)$ for all edges, where e indicates the number of edges in the edge list. Hence, the total complexity is $O(n^2 + e)$.

Algorithm 1: Core Node Selection

Input: $\mathcal{G}$, $\mathcal{C}$, δ
Output: $\mathcal{N}_c$

```
Extract the edge E from the adjacency matrix A.
foreach e_i ∈ E do
    e_i.weight = sim(v_i, v_j);
end
foreach n_i ∈ N do
    n_i.degree = Σ_{j∈N} A_ij · sim(v_i, v_j);
end
Sort(E)
foreach e_i ∈ E do
    if e_i.weight < δ then
        countine;
    end
    if n_i, n_j ∉ N_c then
        if n_i.degree > n_j.degree then
            N_c = N_c ∪ n_i
        end
        else
            N_c = N_c ∪ n_j
        end
    end
end
```

4.3.3. Reformulating Constraints. We then reformulate constraints in the OPT-2 problem into solvable forms.

Prediction Loss. The prediction loss constraint (5) requires that the largest element in the prediction vector be unaltered after adding the noise vector. Hence, it can be rewritten as a set of inequalities: $v^* + s^* - (v^a + s^a) \geq 0, \forall v^a \in \mathbf{v}$, where v^a and s^a respectively represent entries in the prediction vector and in the noise vector, while v^* and s^* refers to the largest element in the prediction vector $\mathbf{v}$ and the corresponding entry in the noise vector s. We require the element $v^* + s^*$ to remain larger than all other entries.

Algorithm 2: Iterative Gradient Descent for OPT-GRID with KKT Conditions

Input: Initial noise vector $s^{(0)}$, learning rate α, tolerance ϵ, maximum iterations K, step size for Lagrange multipliers β
Output: Optimized noise vector s
Initialize $s^{(0)}$, $\lambda^{(0)}$, $\mu^{(0)}$, $\nu^{(0)}$;
for $k = 0$ **to** K **do**
  Calculate $\nabla_s \mathcal{L}$;
  $s^{(k+1)} = s^{(k)} - \alpha \nabla_s \mathcal{L}$;
  $s^{(k+1)} = \text{ConstraintCheck}(s^{(k+1)}, v_i, \theta)$;
  Update $\lambda^{(k+1)}$, $\mu^{(k+1)}$, $\nu^{(k+1)}$ using:

$$\lambda^{(k+1)} = \max\left(0, \lambda^{(k)} + \beta\left(v^a + s^a - v^* - s^*\right)\right)$$
$$\nu^{(k+1)} = \max\left(0, \nu^{(k)} + \beta\left(\|\mathbf{s}\| - \theta\right)\right)$$
$$\mu^{(k+1)} = \mu^{(k)} + \beta\left(\sum_a s^a\right)$$

  if $\|s^{(k+1)} - s^{(k)}\| < \epsilon$ **then**
    break
  end
end
return $s = s^{(k+1)}$;

Algorithm 3: ConstraintCheck Handling

Input: Noise vector s, Prediction vector v_i, Norm threshold θ, and Vector dimension A
Output: Constrained noise vector s
Step 1: Handle Distribution Constraint L2:

$$s = s - \frac{1}{A}\left(\sum_{a=1}^{A} s^a\right)\mathbf{1}$$

for $a = 1$ **to** A **do**

$$s^a = \min\left(\max\left(s^a, -v_i^a\right), 1 - v_i^a\right)$$

end
Step 2: Handle Norm Constraint L3:
if $\|\mathbf{s}\| > \theta$ **then**

$$\mathbf{s} = \mathbf{s} \times \frac{\theta}{\|\mathbf{s}\|}$$

end
Step 3: Handle Label Constraint L1:
Let $C \leftarrow \arg\max_a v_i^a$;
for $a = 1$ **to** A **do**
  if $a \neq C$ ***and*** $v_i^C + s^C - (v_i^a + s^a) < 0$ **then**
    $\delta \leftarrow \frac{v_i^a + s^a - v_i^C - s^C}{2}$;
    $s^a \leftarrow s^a - \delta$;
    $s^C \leftarrow s^C + \delta$;
  end
end
return s;

Probability Distribution. The prediction loss constraints ensure that the noisy vector follows a normal distribution. The first requirement, $0 \leq v^a + s^a \leq 1$, can be directly treated as an inequality constraint. The second requirement, $\sum_a (v^a + s^a) = 1$, can be simplified as $\sum_a (s^a) = 0$, since $\sum_a (v^a) = 1$.

Distortion Budget. The distortion budget constraint (7) controls the difference between the original prediction vector and the distorted prediction vector. In this paper, we adopt the L-1 norm to measure the difference between two vectors, so this constraint can be rewritten as $\|\mathbf{s}\| - \theta \leq 0$, which represents that the sum of the absolute value of each element in the noise vectors should be less than the distortion budget.

4.3.4. Reformulation of OPT-GRID. Finally, we summarize all efforts to reformulate OPT-2 into the OPT-GRID. Given the prediction vectors of all nodes in the graph, for each node in the core node set, $n_i \in \mathcal{N}_c$, we calculate its noise vectors s by solving the following optimization problem.

$$\begin{aligned}
\text{OPT-GRID: } \min_{s} \; & D_i \\
s.t. \; & \text{L1: } (v^a + s^a) - (v^* + s^*) \leq 0, \forall v^a \in \mathbf{v_i} \; ; \\
& \text{L2: } \sum_a (s^a) = 0 \; ; \\
& \quad 0 \leq v^a + s^a \leq 1 \; ; \\
& \text{L3: } \|\mathbf{s}\| - \theta \leq 0 \; .
\end{aligned}$$

In this optimization problem, the optimized variable is the noise vector s, which impacts the calculation of the similarity gap D_i according to Eqn. (8). s^* represents the entry in the noise vector $\mathbf{s}$ where the corresponding v^* is the largest element in the prediction vector $\mathbf{v_i}$. OPT-GRID is a constrained non-linear optimization problem, known to be difficult to solve for a global optimal solution in polynomial time, especially considering the training graph with a large size and complex topology. To address this problem, we apply the method of Lagrange multipliers, transforming the original constrained problem into an unconstrained one with a Lagrange function, as follows:

$$\mathcal{L} = D_i + \lambda(v^a + s^a - v^* - s^*) + \mu \sum_a (s^a) + \nu(\|\mathbf{s}\| - \theta) \; .$$

Note that, the constraint $0 \leq v^a + s^a \leq 1$ is not appended with the Lagrange multiplier because it represents simple bound constraints that can be efficiently enforced through projection to improve computational efficiency. Including this constraint in the Lagrangian would require introducing additional Lagrange multipliers for each bound, which is unnecessary and would complicate the optimization process without providing significant benefits. Moreover, we use a single Lagrangian multiplier λ for the L1 constraint to simplify the formula expression and enhance readability. In practice, a unique Lagrangian multiplier λ^a is assigned for each $v^a \in \mathbf{v}_i$, ensuring that the L1 condition is satisfied for every v^a. We then employ the Karush–Kuhn–Tucker

(KKT) conditions to regulate the optimization process. The KKT conditions include primal feasibility (the original constraints) with dual feasibility, complementary slackness, and stationarity, as shown below:

$$\lambda \geq 0,\ \nu \geq 0;$$
$$\lambda(v^a + s^a - v^* - s^*) = 0,\ \nu(\|\mathbf{s}\| - \theta) = 0;$$
$$\nabla_s \mathcal{L} = 0\ .$$

The calculation of $\nabla_s \mathcal{L}$ can refer to Appendix A.2. For each node in the core node set, we adopt the gradient descent method to iteratively update the noise vector, as outlined in Algorithm 2. The Lagrange multipliers are updated using the subgradient method to satisfy the dual feasibility and complementary slackness conditions of the KKT conditions and we control the adjustment magnitude to maintain stability and convergence. More specifically, for λ and ν, the update increases their value when the constraint is violated and vice versa. By updating them based on the constraint violations, the algorithm moves toward satisfying the complementary slackness conditions. For μ, the update adjusts its value based on the sum of $\sum_a s^a$, to ensure that $\sum_a s^a$ can move toward zero over iterations. Besides, the function ConstraintCheck$(s^{(k+1)}, v_i, \theta)$ ensures that the primal feasibility is satisfied by adjusting the solution $s^{(k+1)}$, with the details illustrated in Algorithm 3. Especially, the constraint $0 \leq v^a + s^a \leq 1$ is enforced through the projection $s^a = \min\left(\max\left(s^a, -v_i^a\right), 1 - v_i^a\right)$. In this adjustment, $\max\left(s^a, -v_i^a\right)$ ensures that s^a is always no less than $-v_i^a$, i.e., $0 \leq s^a + v_i^a$ Besides, $\min\left(s^a, 1 - v_i^a\right)$ ensures that s^a is always no greater than $1 - v_i^a$, i.e., $v_i^a + s^a \leq 1$.

Once the update process converges, as indicated by changes in the noise vector smaller than η, or reaching the maximum iterations K, the algorithm terminates. Following this process, we can find a near-optimal solution for the noise vector of each node in the core node set and then append them to the prediction vectors of the target model.

5. Performance Evaluation

We conduct experiments to assess GRID's performance in defending different types of link-stealing attacks across various datasets. Second, we compare GRID to its counterparts in terms of defending against link-stealing attacks.

5.1. Experiment Settings

5.1.1. Datasets. We evaluate our approach on five widely adopted graph datasets: Citeseer [42], Cora [42], Pubmed [42], AIDS [5], and ENZYMES [9], which are commonly used for assessing GNN models.

5.1.2. Link Stealing Attacks. Our evaluations primarily focus on the first link-stealing attacks [11], as they provide the general and comprehensive framework for covering various attack scenarios. Among the eight proposed attacks, we highlight the performance of three representative attacks that encompass two key perspectives: unsupervised and supervised learning, as well as black-box and gray-box settings. For the unsupervised Attack-0, we employ AUC (area under the ROC curve) as the primary evaluation metric to evaluate their performance. Additionally, to demonstrate the attack accuracy, recall, and precision metrics more concretely, we adopt the K-means clustering approach as implemented in [11]. We set the value of K to 2, where the cluster with a lower (higher) averaged distance value is considered as the set of adjacent (non-adjacent) node pairs. For the supervised black-box Attack-1 and supervised gray-box Attack-6, we follow their original settings and utilize their source codes [2] to implement the attack and calculate the AUC, accuracy, recall, and precision with and without incorporating our GRID defense mechanism. Additionally, in Section 5.3 and 5.4, we also include the other five attacks proposed in [11] to ensure a comprehensive evaluation. Second, we also consider the latest attack [46], i.e., ILS, which is designed toward the inductive GNN model. Specifically, we select attacks A2 and A9, which were reported as the strongest in the posterior-only and combined attack categories, to demonstrate our GRID performance. The details about these attacks, including the setting of the shadow model and the link stealing classifiers, can be referred to [11] and [46].

5.1.3. Dataset Configurations. Regarding AIDS and ENZYMES, which are designed for graph classification, we assign the labels of the graph (molecular compounds) to nodes (molecules) as their ground truth labels. During our experiments, we need to train a target model, implement the link stealing attacks, and evaluate the defense method, so we extract several disjointed sub-datasets from the original dataset D for experiments, depicted below.

D1 (for training the target model). We sample 40% of the nodes and of the links from D to compose the training graph $D1$. For the transductive training, e.g., GCN, we retained the full graph structure in D1 but removed 20% nodes' labels. The model was then trained to predict the labels of these unlabeled nodes. For the inductive training, e.g., GAT, we trained the model directly on D1 and let it predict unseen nodes in D.

D2 (for performing attacks). Some link-stealing attacks (i.e., Attack-1, Attack-4, Attack-5, and Attack-7) require a shadow dataset to perform the attack. We sample 40% of the nodes and the corresponding links from D to compose the shadow training graph $D2$. It will be employed to train the shadow model for these attacks. Note that not all the data samples in D2 are adopted to train the shadow model. The detailed adoption ratio can refer to [11] and [2].

D3 (for other defenses). Existing defense methods may require the model provider to train a defender classifier to fit the attacker's inference classifier. So, we sample 20% of the nodes and the corresponding links from D to compose the graph $D4$ for the defender classifier.

D4 (for testing attacks). In D1, we sample 20% node pairs that are linked and another 20% node pairs that are unlinked

TABLE 2. LINK STEALING ATTACK-0 PERFORMANCE (%) ON FIVE DATASETS WITH AND WITHOUT OUR GRID DEFENSE. NOTE: 'A/B' ARE THE VALUES UNDER GCN AND GAT

	Citeseer		Cora		Pubmed		AIDS		ENZYMES	
	No Defense	GRID	No Defense	GRID	No Defense	GRID	No Defense	GRID	No Defense	GRID
Attack Accuracy	86.7 / 86.9	65.8 / 66.1	85.2 / 85.1	63.2 / 62.8	78.5 / 78.8	61.4 / 61.8	75.1 / 74.7	60.1 / 60.5	74.2 / 73.8	57.9 / 57.6
Attack Precision	77.6 / 77.4	66.1 / 66.4	76.8 / 76.5	63.2 / 63.6	69.4 / 69.6	57.2 / 57.6	52.7 / 52.3	51.0 / 50.7	53.1 / 53.6	51.4 / 51.0
Attack Recall	98.1 / 97.7	64.3 / 64.7	95.7 / 96.1	63.1 / 63.3	94.5 / 94.2	63.2 / 62.8	97.6 / 97.3	64.5 / 64.8	98.7 / 98.4	63.3 / 63.1
Attack AUC	94.2 / 93.8	69.1 / 68.8	93.0 / 92.6	68.5 / 68.8	86.9 / 86.5	67.5 / 67.8	70.1 / 70.5	59.2 / 59.6	63.1 / 63.4	56.4 / 56.1
Model Accuracy	73.3 / 75.8	73.3 / 75.8	84.8 / 85.6	84.8 / 85.6	81.6 / 83.3	81.6 / 83.3	68.5 / 70.5	68.5 / 70.5	69.0 / 71.4	69.0 / 71.4

TABLE 3. LINK STEALING ATTACK-1 PERFORMANCE (%) ON FIVE DATASETS WITH AND WITHOUT OUR GRID DEFENSE. NOTE: 'A/B' ARE THE VALUES UNDER GCN AND GAT

	Citeseer		Cora		Pubmed		AIDS		ENZYMES	
	No Defense	GRID	No Defense	GRID	No Defense	GRID	No Defense	GRID	No Defense	GRID
Attack Accuracy	90.1 / 89.7	67.5 / 67.8	87.2 / 87.5	66.3 / 66.0	85.1 / 84.7	62.4 / 62.7	71.5 / 71.8	55.0 / 55.3	69.0 / 69.3	53.2 / 53.5
Attack Precision	87.1 / 86.9	67.1 / 67.4	85.4 / 85.2	66.8 / 66.5	77.5 / 77.3	62.1 / 62.4	72.5 / 72.2	55.2 / 55.0	68.3 / 68.0	54.1 / 54.3
Attack Recall	95.8 / 95.5	68.4 / 68.2	88.3 / 88.0	65.8 / 66.1	89.1 / 89.4	62.6 / 62.4	70.1 / 70.3	54.7 / 54.5	70.1 / 70.3	52.7 / 53.0
Attack AUC	96.5 / 96.3	73.2 / 73.5	94.2 / 93.9	72.8 / 73.1	88.5 / 88.2	70.6 / 70.4	72.9 / 73.1	58.9 / 58.7	74.5 / 74.2	60.2 / 60.0
Model Accuracy	73.3 / 75.8	73.3 / 75.8	84.8 / 85.6	84.8 / 85.6	81.6 / 83.3	81.6 / 83.3	68.5 / 70.5	68.5 / 70.5	69.0 / 71.4	69.0 / 71.4

TABLE 4. LINK STEALING ATTACK-6 PERFORMANCE (%) ON FIVE DATASETS WITH AND WITHOUT OUR GRID DEFENSE. NOTE: 'A/B' ARE THE VALUES UNDER GCN AND GAT

	Citeseer		Cora		Pubmed		AIDS		ENZYMES	
	No Defense	GRID	No Defense	GRID	No Defense	GRID	No Defense	GRID	No Defense	GRID
Attack Accuracy	92.2 / 92.3	69.1 / 69.4	89.9 / 90.1	69.5 / 69.3	91.1 / 90.9	68.8 / 68.6	94.3 / 94.2	68.2 / 68.3	82.4 / 82.6	65.0 / 64.9
Attack Precision	90.1 / 90.0	69.1 / 68.9	87.8 / 87.6	69.7 / 69.6	90.3 / 90.2	69.7 / 69.5	90.7 / 90.8	69.5 / 69.6	77.0 / 76.9	65.3 / 65.1
Attack Recall	93.3 / 93.5	69.2 / 69.0	93.0 / 93.2	69.3 / 69.5	92.4 / 92.3	68.5 / 68.7	98.6 / 98.5	67.5 / 67.6	88.7 / 88.5	64.8 / 64.9
Attack AUC	98.1 / 98.2	71.1 / 71.0	96.4 / 96.3	70.4 / 70.5	97.0 / 96.9	70.8 / 70.7	97.9 / 97.8	73.9 / 74.0	89.1 / 89.0	68.1 / 68.3
Model Accuracy	73.3 / 75.8	73.3 / 75.8	84.8 / 85.6	84.8 / 85.6	81.6 / 83.3	81.6 / 83.3	68.5 / 70.5	68.5 / 70.5	69.0 / 71.4	69.0 / 71.4

TABLE 5. THE PERFORMANCE (%) OF TWO ILS ATTACKS, A2 AND A9, ON GAT ACROSS FIVE DATASETS, WITH AND WITHOUT OUR GRID. A2 AND A9 ARE REPORTED AS THE STRONGEST IN THE POSTERIOR-ONLY AND COMBINED ATTACK CATEGORIES, RESPECTIVELY

		Citeseer		Cora		Pubmed		AIDS		ENZYMES	
		No Defense	GRID	No Defense	GRID	No Defense	GRID	No Defense	GRID	No Defense	GRID
ILS A2	Accuracy	82.54	66.21	83.71	67.41	82.71	64.27	75.44	60.68	75.51	60.01
	Precision	82.03	66.87	83.15	68.22	83.28	64.51	76.03	60.34	74.79	60.35
	Recall	82.85	66.76	82.87	67.04	83.12	64.79	75.08	61.13	75.94	60.55
	AUC	82.51	60.60	84.50	61.02	80.92	57.01	82.72	57.81	81.24	57.90
ILS A9	Accuracy	92.45	67.53	91.76	68.84	90.67	66.58	84.47	63.58	82.50	64.01
	Precision	91.96	67.92	91.25	69.44	90.51	67.24	84.05	63.79	83.00	64.59
	Recall	92.72	66.98	91.85	68.31	91.14	65.07	84.80	63.20	82.21	63.06
	AUC	90.98	67.01	91.05	64.04	93.89	64.17	84.07	63.42	85.68	61.26
Model Accuracy		75.80	75.80	85.60	85.60	83.30	83.30	70.50	70.50	71.41	71.41

to compose $D4$, for use to evaluate the link stealing attacks with and without defense.

5.1.4. Target Model. We consider both transductive and inductive GNN models. For the transductive one, we train the model of graph convolutional networks (GCN) for the node classification task, as described in [16]. Besides, for inductive models, we focused on GAT [38], GraphSAGE [10], and GIN [47]. We train the target model with D1 and sample another 20% nodes in D to evaluate model performance. We implemented all these models based on publicly accessible code [1], [3]. The testing accuracy outcomes of target models on each dataset are similar to the results presented in the benchmark. Our experiments are conducted on a workstation equipped with Intel Core i9-13900K and NVIDIA RTX 4090 GPU with 24GB of VRAM.

5.2. Overall Performance on Different Attacks

We employ our GRID to show defense performance results for different attack types under different models and datasets. Here, we set the distortion budget $\theta = 0.4$ and the hop $n = 3$. It is important to note that in our context, a noise level of $\theta = 0.4$ represents a relatively minor perturbation. For example, $(-0.2, 0.2)$ is the largest noise for $\theta = 0.4$ in the two-dimensional noise vector. Given that the prediction vectors for the target model exist in higher dimensions, such as 3 for Pubmed, 6 for Citeseer, and 7 for Cora, the resulting distortion on each confidence score remains quite minimal.

Regarding the first similarity-based attacks [11], the results for Attack-0, Attack-1, and Attack-6 on GCN and GAT are shown in Tables 2, 3, and 4. Additional results on GraphSAGE and GIN can be found in Appendix A.4. It is important to note that GRID does not degrade the target model accuracy, as shown in the last row of these tables. The

TABLE 6. ATTACK ACCURACY AND COMPUTATIONAL TIME UNDER GRID WITH AND WITHOUT CORE NODE SELECTION. NOTE: 'A/B' ARE THE VALUES UNDER GCN AND GAT

	w/ Core Node Selection				w/o Core Node Selection
	n=2	n=3	n=4	n=5	
Attack Accuracy					
Attack-0	62.7 / 63.6	61.4 / 61.8	61.3 / 61.3	60.2 / 60.8	68.3 / 69.8
Attack-1	63.4 / 64.2	62.4 / 62.7	61.8 / 61.6	60.4 / 59.7	68.4 / 68.8
Attack-2	62.4 / 62.1	61.4 / 61.5	60.1 / 60.8	59.4 / 58.5	67.6 / 69.9
Attack-3	64.0 / 63.9	63.3 / 63.3	62.0 / 61.9	60.2 / 60.1	68.4 / 68.7
Attack-4	64.5 / 64.3	63.5 / 64.0	61.2 / 61.2	59.5 / 60.0	69.0 / 68.9
Attack-5	67.9 / 66.8	65.9 / 66.2	63.5 / 64.0	62.0 / 61.8	70.1 / 71.9
Attack-6	69.7 / 69.9	68.8 / 68.6	65.8 / 66.0	63.8 / 64.1	74.5 / 75.6
Attack-7	65.8 / 66.2	64.9 / 65.3	62.9 / 63.0	60.9 / 61.8	70.7 / 71.4
Computation Time	6403.41s / 6391.24s	6823.70s / 6779.71s	7431.28s / 7558.92s	7937.75s / 7979.18s	45052.53s / 46148.31s

prediction loss constraint in our optimization formulation ensures that the target model accuracy remains unchanged under GRID. Furthermore, our noise vector calculation does not require re-training the target model, so the training loss and expense remain unaffected.

First, Table 2 shows the performance of unsupervised Attack-0 under our GRID, where our GRID effectively defends against this attack across five datasets. The evaluation of unsupervised attacks heavily relies on the AUC metric, and a significant degradation of AUC scores was observed when our GRID was deployed. Notably, in the case of three citation datasets where attacks initially achieved superior performance, our GRID can incur an approximate 25% degradation in their AUC metrics. Even for the two chemical datasets where attacks exhibited subpar performance, our GRID can reduce their AUC to around 60%, rendering the corresponding attacks to random guessing. Second, we turn our attention to the black-box attack, i.e., Attack-1, as depicted in Table 3, which showcases the performance variation under the influence of our GRID. The attack performance exhibits a significant degradation across all assessed metrics under the defense of our GRID. Across the three citation datasets, we observed a degradation of approximately 20% in all metrics, with a particularly noticeable deterioration of 25% in the recall. For the two chemical datasets, our GRID exhibits better performance, reducing the attack performance to almost 50%. Such an attack performance is only marginally better than random guessing. This significant reduction in attack performance underscores the effectiveness of our GRID in defending against black-box attacks. Third, regarding the strongest gray-box attacks, Attack-6, our GRID performs a little bit inferior. The reason is that these attacks can leverage the certain knowledge of node attributes and graph topology to launch strong attacks, while our GRID primarily focuses only on the prediction vectors. However, our GRID is still able to reduce its AUC performance to around 70% with a distortion budget of only 0.4.

Additionally, the results of A2 and A9 from the latest ILS attack [46] on GAT are shown in Table 5. To favor the attacks, the shadow models used to perform the attacks are identical to the target model. Despite this, we observe that our GRID method still significantly reduces the performance

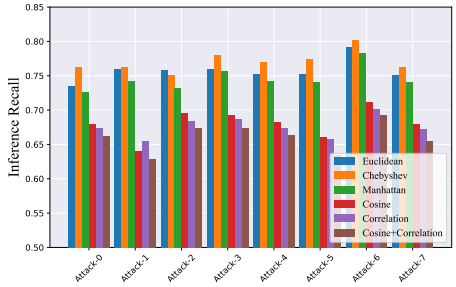


Figure 1. Link stealing attack recall variations for eight different attacks under different similarity metrics.

of various ILS attacks. This reason is that ILS also relies on similarity metrics calculated from the prediction vectors of pairwise nodes. By adding noise to the prediction vectors of the training nodes output with our GRID, the similarity patterns were distorted, thereby misleading the attack classifiers.

5.3. Ablation Study

Impact of Core Node Selection. The core node selection is a key design element in GRID that addresses computational complexity and noise offset issues. Therefore, we compare our GRID with and without this component to analyze its contribution to defense performance. For our GRID method with core node selection, we set n between 2 and 5, and the noise is calculated only for the nodes in the core node set. Besides, the thresholds δ, determined by the similarity of the prediction vectors of n-hop indirect neighboring nodes, are approximated using a maximum of 1,000 pairs of n-hop nodes. For GRID without core node selection, the noise is calculated for all nodes individually. The distortion budget is set to 0.4, and the maximum iteration for solving the optimization problem is set to 20. Here, we take Pubmed with 19717 nodes as an example, and 3201 nodes are chosen as the core node.

The results of computational time and the defense performance of eight attacks in [11] are shown in Table 6. First, we observe that the defense performance of GRID with core node selection is consistently better, as indicated by the lower attack accuracy across all eight attacks. This improvement is because, under GRID without core node

selection, the noise for the two end nodes of each link tends to counteract during similarity disguise, as each noise is optimized for each node's neighboring nodes individually. Second, the computational time for GRID with core node selection, which is less than 8,000 seconds, is significantly lower than that for GRID without core node selection, which exceeds 45,000 seconds. The computational time of GRID is highly dependent on the number of nodes selected for noise calculation. Under the core node selection scheme, only a subset of nodes needs to be calculated, compared to the total 19,717 nodes considered in GRID without core node selection. Additionally, since we sample a maximum of 1,000 pairs of n-hop nodes, the computational time for the similarity threshold calculation does not significantly increase as n increases. These two factors together result in substantial savings in computational time. Furthermore, as n increases, we observe a slight improvement in defense performance, although computation time also increases due to more nodes being selected as core nodes. This observation underscores the importance of selecting n in practice to balance defense performance with computational time.

Impact of Different Similarity Metrics. Our current GRID design relies on the combination of correlation and cosine similarity measurement due to their superior performance in executing link-stealing attacks as reported in [11]. Here, we adopt different similarity metrics and show their impacts on our GRID. In particular, the Euclidean, Chebyshev, Manhattan, Cosine, and Correlation are taken into account. That is, we replace our similarity calculation with each of the aforementioned metrics and evaluate the effectiveness of our GRID. Here, we set the distortion budget $\theta = 0.3$, the $n = 3$, and conduct experiments on the Citeseer dataset. Our results are shown in Fig. 1.

As observed in Fig. 1, the combination of correlation and cosine consistently outperforms all other individual metrics across all attacks, which decreases the recall values at the maximum. For instance, in the case of Attack-6, the recall value for the combination of correlation and cosine is 0.692, while the recall values for Euclidean, Chebyshev, Manhattan, Cosine, and Correlation metrics are 0.7921, 0.801, 0.7831, 0.712, and 0.701, respectively. On the other hand, we find that our defense performance under the Correlation or Cosine is always much better than that under other individual metrics. This further enhances our confidence to take the combination of correlation and cosine similarity rather than other individual or combined metrics.

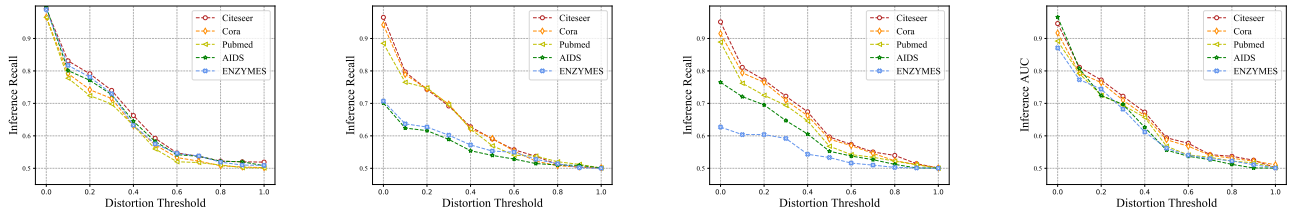
5.4. GRID Performance under Different Settings

Recall that there are two important control parameters for GRID: the distortion budget θ, which controls the maximum distortion incurred by noise, and the value n, indicating the number of hops for indirect neighboring nodes, which influences both the similarity gap in the objective function and the threshold δ for core node selection. Here, we analyze how these parameters impact our GRID.

5.4.1. Defense Performance on Different Distortion Budget θ. The distortion budget θ represents the upper bound of the difference between the original prediction vector and the distorted one. A larger budget signifies that the model provider can tolerate a larger distortion caused by the noise. Here, we set the hop $n = 3$ and vary θ from 0 to 1 with the step of 0.1. Since recall can measure the critical ability of whether an attacker can extract all links from the training graph, we will use recall as the primary metric to demonstrate our defense performance. Figures 2(a) to 2(h) respectively show the recall variations regarding the attack performance of eight attacks on five datasets under the defense of our GRID when varying the distortion threshold θ on GCN. We have the following observations.

Our GRID capability is enhanced with the increase in the distortion budget. From Figures 2(a) to 2(h), we observe that the recall values of all eight link stealing attacks drop with an increase in the distortion budget. In particular, all recall values on five datasets drop to around 50% (similar to random guessing) when the distortion budget approaches 1.0. Taking the black-box Attack-1 as an example, i.e., Figure 2(b), when θ increases from 0 to 1, the recall values for Attack-1 drop from 96.5% to 50.1%, from 94.2% to 50.3%, from 88.5% to 50.1%, from 70.1% to 50.2%, and from 70.7% to 50.2%, respectively, on five datasets. Similarly, for the strongest attack, i.e., Attack-6, the recall values of attack performance drops from 95.1% to 50.9%, from $84.4.8\%$ to 50.2%, from 85.0% to 50.9%, from 97.9% to 50.3%, and from 89.1% to 50.4%, respectively, on five datasets, as shown in Figure 2(g). The rationale behind this is that a larger distortion threshold allows more substantial noise vectors to be added to the prediction vectors, incurring a noticeable variation in the similarity of altered prediction vectors from two adjacent nodes. That is, the prediction similarity level between two adjacent nodes is akin to that of 3-hop indirect neighboring nodes, confusing link-stealing attacks.

GRID can incur quick decay for attack performance even with minor noise. We also observe that even with a small distortion budget i.e., $\theta = 0.1$, the recall values still drop quickly for all attacks on all datasets. For example, the recall values drop by 16.01%, 16.80%, 14.01%, 13.56%, 12.97%, 12.8%, 11.9%, and 13.6% respectively on the Citesser dataset across all eight attacks. This can deteriorate the attack performance from approximately 95% to below 80%. This observation demonstrates that our GRID can achieve notable defense performance even with negligible distortion on the prediction vectors. The reason can be attributed to the fact that the link stealing attacks rely on, and are sensitive to, the similarity difference between adjacent nodes and non-adjacent nodes. Our GRID can effectively reduce such differences with slight noise, for hiding the adjacent nodes into the indirect neighboring nodes to thwart the attacks. In addition, we notice that when θ increases to 0.4, the recall values drop to around 60% for all attacks. Hence, a suggested θ value of 0.4 is well enough to defend against all link-stealing attacks.



(a) Attack-0 (b) Attack-1 (c) Attack-2 (d) Attack-3

(e) Attack-4 (f) Attack-5 (g) Attack-6 (h) Attack-7

Figure 2. Link stealing attack recall variations for eight different attacks under GRID with increasing distortion thresholds.

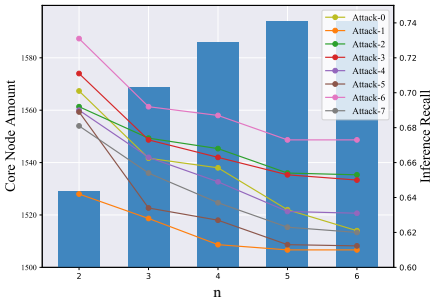


Figure 3. Link stealing attack recall and the core node set size variations for eight different attacks with increasing n.

5.4.2. The Impact of n-hop. The n value will affect the similarity gap in the objective function and the threshold δ for core node selection. Here, we set the distortion budget $\theta = 0.3$ and vary n from 2 to 6, to show the link stealing attacks' recall values and the core node set size variations. We conduct experiments for the Citesser dataset on GCN as the example, with the results shown in Fig. 3. We observe that when n increases, the node amounts in the core node set rise marginally while the recall values appear to drop. For instance, when n increases from 2 to 6, the number of core nodes increases from 1515 to 1569, 1586, 1594, and 1596. The recall values corresponding to eight attacks drop from 0.701 to 0.621, 0.642 to 0.610, 0.692 to 0.653, 0.711 to 0.650, 0.690 to 0.631, 0.689 to 0.612, 0.7311 to 0.673, and 0.681 to 0.620, respectively. The reason is that when n is larger, there are fewer correlations for a node and its n-hop nodes, resulting in smaller values of δ. Hence, fewer links will be discarded in Algorithm 1, leading to an increased number of selected core nodes. Meanwhile, our GRID will enforce the similarity between adjacent nodes to the less correlated nodes, which thus can better degrade the attack performance. However, this requires generating larger similarity distortion, which may incur large distortion in prediction vectors. Besides, as the results are shown in Table 6, the increase in n will also increase the computation time as more noises need to be calculated. Hence, a careful selection of n is required to balance defense performance, distortion loss, and computation time.

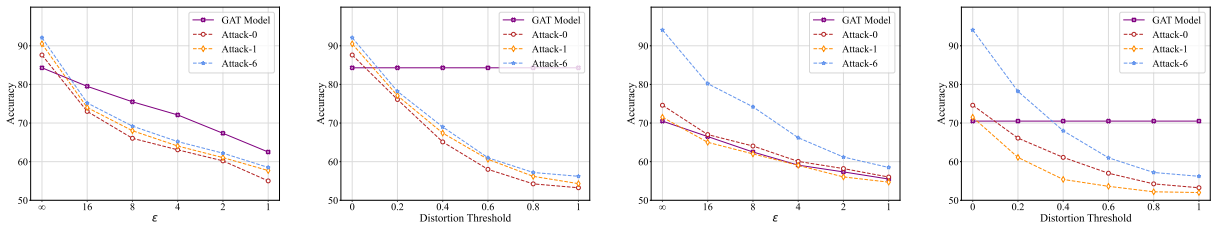
5.5. Comparative Results

We compare our approach with the prominent defense solutions against general inference attacks, as well as DP-based approaches specifically designed to defend against link-inference attacks.

5.5.1. Counterparts. Five state-of-the-art defense methods are taken as our counterparts, which are L2-Regularizer [32], Dropout [36], Min-Max Game [23], MemGuard [15], and the DP-based method, GAP [26]. Among them, L2-Regularizer, Dropout, Min-Max Game, and MemGuard were originally designed to defend against membership inference attacks. GAP was a novel DP-based GNN method, reported to achieve the best performance among DP-based defenses [26] The method details are illustrated in Appendix A.3.

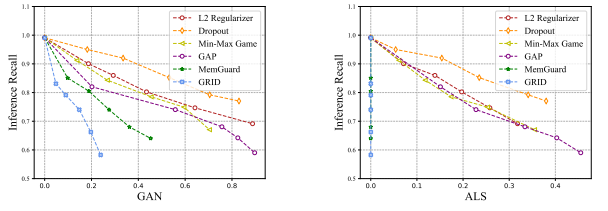
5.5.2. Defense Settings and Metrics. To compare the performance of different defense methods, we will show the attack's recall-utility loss curve with the defense of each method, to demonstrate the trade-off between the defense performance and the utility loss.

Hyperparameter Configurations. For the L2-Regularizer, Dropout, Min-Max Game, and GAP, we first train the target model without defense to record the prediction vectors for some testing samples. Then, we equip each defense under each hyperparameter to retrain the target model and record the new prediction vectors for the same testing samples. By comparing the prediction vectors from the model with and without defense, we can characterize different utility losses on these prediction vectors. Meanwhile, we perform the link-stealing attacks on the target model to obtain the corresponding attack recall values. We record the pairs of

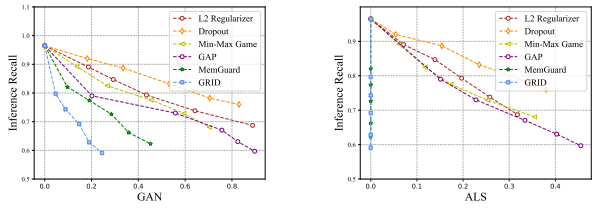


(a) GAP on GAT with Cora (b) GRID on GAT with Cora (c) GAP on GAT with AIDS (d) GRID on GAT with AIDS

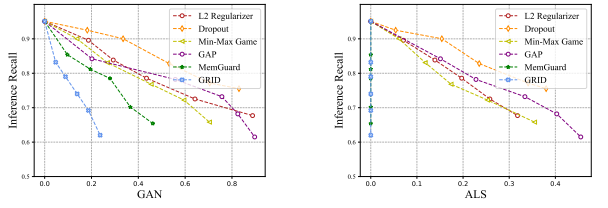
Figure 4. The model prediction accuracy and link stealing attack accuracy variations for GAP and GRID with different hyperparameters.



(a) GAN-Citeseer for Attack-0 (b) ALS-Citeseer for Attack-0



(c) GAN-Citeseer for Attack-1 (d) ALS-Citeseer for Attack-1



(e) GAN-Citeseer for Attack-6 (f) ALS-Citeseer for Attack-6

Figure 5. Trade-offs between the defense performance and the utility loss, including graph-averaged noise and averaged label loss, for Attack-0, Attack-1, and Attack-6 on Citeseer.

attack recall and utility loss under different hyperparameter values to draw the curve. For MemGuard, under each distortion budget, we calculate the noise vectors for all prediction vectors. Then we perform the link-stealing attack on these distorted vectors and calculate the utility loss to obtain the pairs of attack accuracy and utility loss values. Similar to MemGuard, our GRID controls the distortion budget to calculate the noise vectors for the core nodes' prediction vectors and performs the link-stealing attack to obtain the pairs of attack recall and utility loss values.

We sample six trade-off points under each defense method. For L2-Regularizer, the hyperparameter is in the range [0, 0.05] with a step size of 0.01; for Min-Max Game, the hyperparameter is in the range [0, 2.5] with a step size of 0.5; and for Dropout, the hyperparameter is in the range [0, 0.8] with a step size of 0.15. For GAP, the hyperparameter ϵ is set as $[\infty, 16, 8, 4, 2, 1]$, while ∞ represents no defense. For both MemGuard and our GRID, the distortion budget is in the range of [0, 0.5] with a step size of 0.1.

Metrics. We consider two metrics to characterize the utility loss on the prediction vectors resulting from defense methods corresponding to each hyperparameter value. The first one is the graph-averaged noise (**GAN**), which measures the average difference between the original prediction vector and the distorted prediction vector for all nodes in the training graph, calculated by

$$\textbf{GAN}: \Big(\sum_{n_i \in \mathcal{N}_c} d(v_i, v_i + s_i) \Big) / |\mathcal{N}| \ ,$$

where $\mathcal{N}$ is the node set, Notably, **GAN** characterizes the extent of distortions bringing to the target model. The second one is the averaged label loss (**ALS**), which is the fraction of the nodes whose prediction labels, according to their prediction vectors, are altered after employing the defense mechanisms, defined as

$$\textbf{ALS}: \Big(\sum_{n_i \in \mathcal{N}} \mathbb{1}(\arg\max\{\mathbf{v_i}\}, \arg\max\{\mathbf{v_i^*}\}) \Big) / |\mathcal{N}| \ ,$$

where $\mathcal{N}$ is the node set, $\mathbf{v_i}$ and $\mathbf{v_i^*}$ are the prediction vectors with and without defense, and $\mathbb{1}()$ equals 1 if the two terms are the same and equals 0, otherwise.

5.5.3. Comparative Results. It's important to note that to cover five datasets, eight attacks, four models, and two metrics, a total of 320 groups of results would be needed. Thus, we only show some results from certain datasets and models while selecting Attack-0, Attack-1, and Attack-6 in [11] as examples to provide comparative results. Figure 5 presents the trade-off results between defense performance and the utility loss on Citeseer for GRID and its counterparts. From these figures, we observe GRID always achieves the best trade-offs under both attacks: with its curves consistently residing at the leftmost.

Figures 5(a), 5(c), and 5(e) show the trade-offs between defense performance and graph-averaged noise (GAN) on the Citseer dataset under GRID and its five counterparts. Recall that GAN measures the total amount of noise distortion involved in the prediction vectors of all nodes in the training graph. Taking Attack-1 in Figure 5(c) as an example, five existing defense methods lower the attackers' recall values to below 70% at GAN values of 0.868, of $Null$ (dropout can't achieve this performance), of 0.704, of 0.825, and of 0.359, respectively. In contrast, our GRID achieves

the same defense performance at the GAN value of only 0.147. This indicates that our GRID requires much less noise to be added to the target model in order achieve the same defense goal. On the other hand, considering a smaller GAN value, say 0.2, our GRID lowers the recall value of Attack-1 to 62.80% while other methods see their recall values to stand at 89.07%, 92.01%, 89.21%, 79.01%, and 82.01%, respectively. Such results exhibit that GRID can achieve a much better defense performance at the same noise strength. Clearly, GRID outperforms all existing defense mechanisms in terms of the trade-off between the defense goal and introduced noise. The reason lies in the GRID strategy of adding noise vectors exclusively to the prediction vectors of core nodes (1569 out of 3323 for Citeseer). It significantly reduces the total noise amount incorporated in the model.

Figures 5(b), and 5(d), 5(f) draw the trade-off curves between defense performance and the averaged label loss under our GRID and five counterparts. We observe that both our GRID and MemGuard achieve nil label loss, while the other three defense methods exhibit considerable label losses to achieve a given defense performance. The reason is that GRID and MemGuard regulate the label loss constraints to enforce that no prediction label is altered when adding noise vectors. However, the other four methods will involve additional operations in the training process, i.e., calculating the regulation loss, turning off neurons, or adding noise to the data sample, which inevitably compromises their prediction accuracy.

To better compare the privacy-utility tradeoff to DP-based methods, we plot the results showing the variations in model prediction accuracy and link-stealing attack accuracy on GAT under Cora and AIDS, as shown in Figure 4. We observe that, for GAP, a smaller ϵ value offers better defense against attacks but results in significant model accuracy degradation (dropping to 60% at ϵ=1). In contrast, our GRID approach achieves comparable defense performance at a higher distortion threshold, while maintaining zero degradation in model accuracy. Both GRID and DP-based approaches contribute to the privacy-utility tradeoff but from different perspectives. While DP-based methods provide formal privacy guarantees, they interfere with the training process or data, leading to inevitable declines in model performance. On the other hand, our GRID method provides a formal guarantee for model utility without affecting model training or data, although its privacy protection is based on empirical results.

Hence, we conclude that our GRID mechanism achieves superior defense performance in GNNs while incurring the least utility losses than its counterparts examined.

6. Discussion and Future Work

6.1. Defense against Influence-based Attacks

In this paper, we mainly focus on the similarity-based link-stealing attack. But, we also explore our GRID performance in safeguarding the inductive GNN model from influence-based attacks. Considering that influence-based attacks require attackers to inject nodes into the training graph and query the model to get predictions for these nodes, we accordingly tailor GRID to calculate the noises for the prediction vectors of injected nodes each time based on their current neighboring nodes. That is, the attacker can only obtain noisy predictions for the injected nodes.

We perform the latest attacks LinkTeller [45] and Link-Infiltrator [21] on the GAT model as examples. For Link-Teller, we follow their published code and set the hyperparameter k as the accurate graph density to achieve optimal attack performance. For Link-Infiltrator, the threshold is set as e^{-7} as reported in [21]. For our GRID, we set the distortion budget $\theta = 0.4$ and the hop $n = 3$. The results are presented in Table 7. From this, we observe that GRID's performance against influence-based attacks remains robust. Although the recall merely decreases by around 10%, GRID effectively reduces attack precision and AUC to around 60%. Notably, the low precision and AUC in link inference indicate significant misclassification of links between non-neighboring nodes, such as incorrectly identifying connections between individuals who are not acquainted in a social network. The reason is that our GRID method introduces noise to the prediction vectors of the query nodes before and after injecting nodes, consistently causing random variations in their prediction vectors. This misleads attackers into interpreting such variations as the influence of injected nodes transmitted through the links, leading them to predict false positives, thereby degrading the attack's precision. On the other hand, such noise may also cancel the influence caused by the injection node even if two target nodes are linked, causing false negative predictions. However, it is not guaranteed as evidenced by the minor degradation in the attack recall. In our future work, we will consider incorporating massage passing influences into our noise generation tasks to further enhance GRID's performance against influence-based attacks.

6.2. Defense against Adaptive Attacks

In our design of GRID, we have proactively considered the potential scenario that an attacker may know our GRID mechanism and attempt to launch the adaptive attack in that it predicts the presence of a link between two nodes based on the low similarity of prediction vectors of adjacent nodes. To counter such adaptive attacks, our design aims at disguising adjacent nodes into the n-hop indirect neighboring nodes, hence, effectively thwarting such adaptive attacks. Here, we consider another adaptive attack by assuming that the attacker has the knowledge of the certain value of the n-hop. In this attack, an attacker treats all n-hop indirect neighboring nodes as adjacent nodes when constructing the ground truth to train the link-stealing classifier. This allows attackers to learn the similarity patterns of adjacent nodes more effectively. To assess the resilience of our GRID against this adaptive scheme, we utilized Attack-6, which represents the strongest attack, as the backbone for implementing this adaptive strategy. Here, we set the distortion loss $\theta = 0.4$ and $n = 3$ for our GRID.

TABLE 7. INFLUENCE-BASED LINK STEALING ATTACKS PERFORMANCE (%) ON GAT UNDER FIVE DATASETS WITH AND WITHOUT OUR GRID

		Citeseer		Cora		Pubmed		AIDS		ENZYMES	
		No Defense	GRID	No Defense	GRID	No Defense	GRID	No Defense	GRID	No Defense	GRID
LinkTeller	Accuracy	83.65	66.03	84.76	67.84	82.07	63.08	73.54	59.58	71.25	58.20
	Precision	83.63	58.70	83.36	57.92	82.24	56.75	73.05	54.04	71.74	54.72
	Recall	83.95	75.01	85.04	74.77	81.67	73.31	74.81	68.03	70.02	62.75
	AUC	84.70	60.60	83.55	61.02	81.79	57.01	73.57	57.81	72.18	57.90
Link-Infiltrator	Accuracy	91.35	74.53	92.76	75.84	89.97	70.58	80.54	66.58	79.25	63.20
	Precision	91.13	64.20	91.86	64.42	90.24	61.25	81.05	59.54	79.74	56.05
	Recall	91.45	83.51	93.54	83.27	89.67	81.81	82.31	76.53	78.52	69.25
	AUC	92.20	66.10	91.05	66.52	89.29	65.51	82.07	64.31	85.68	64.40
Model Accuracy		75.80	75.80	85.60	85.60	83.30	83.30	70.50	70.50	71.41	71.41

TABLE 8. ADAPTIVE ATTACK PERFORMANCE UNDER OUR GRID

	Citeseer	Cora	Pubmed	AIDS	ENZYMES
Precision	0.591	0.598	0.572	0.589	0.575
Recall	0.772	0.763	0.774	0.723	0.715
Accuracy	0.682	0.681	0.672	0.657	0.647
AUC	0.792	0.793	0.774	0.723	0.715

Table 8 lists our results on five datasets. Comparing to Table 4, we observe that such an adaptive attack has a positive impact on the recall values, boosting them to approximately 75% due to its ability to learn additional similarity patterns of adjacent nodes, thereby improving link recognition. However, it also comes at the cost of deteriorating the attack precision to lower than 60%. This decline occurs since the adaptive attack wrongly identifies n-hop indirect neighboring nodes as adjacent nodes when constructing the ground truth for training the link-stealing classifier. The trade-off between inference recall and precision underscores the robustness of our preventive measure, which disguises the similarity levels of adjacent nodes into the n-hop indirect neighboring nodes. By doing so, GRID can effectively hinder attackers from exploiting such adaptive strategies. However, this attack is merely an example of the heuristic argument; in practice, the attackers may devise other strong adaptive attacks, such as optimization problem-based schemes. Thus, in our future work, we will refine defense formulations and provide a mathematical proof of its guaranteed performance under stronger adaptive attacks.

7. Related Work

Link Inference Attack. The link inference attacks toward GNN models were proposed in [11], aiming at stealing the link of the training graph. It leverages the characteristics of the GNN model, which aggregates information for each node from its adjacent nodes, causing a high similarity in the prediction vectors of adjacent nodes. Hence, an attacker can measure the similarity pattern of two queried nodes to determine whether a link exists between them. Subsequent work employs such a similarity-based method to attack the unsupervised graph representation learning or inductive GNN models [41], [46]. Besides, [45], [21] considered the scenario where the attacker can inject the malicious nodes around the target nodes in the training graph. Then, the attacker can measure the variation of prediction vectors queried for the target node before and after node injection to infer whether they are linked, which is called the influence-based attack.

Membership Inference Attack The membership inference attacks aim to infer if a data point belongs to the training dataset of deep learning. It was first proposed in [32], and then plentiful follow-up research was conducted to improve the original designs or apply them to different learning algorithms/models [29], [19], [17], [8], [20], [28], [22], [25], [34], [35], [13], [6], [33], [18], such as contrastive learning, online learning, transfer learning, federated learning, natural language domains, among others. These attacks are not designed originally for GNN models, so their extension to steal links in the training graph of a GNN remains open. On the other hand, some studies [12], [44], [24] have extended the membership inference attack into the context of GNNs, for inferring whether a node sample was in the training graph. However, the exploration of defending membership inference attacks is out of the scope of our work.

Model Inference Attack. The model inference attacks have been proposed for stealing model parameters in [7], [14], [37], [39]. The attacker typically doesn't have direct access to the target model's architecture, weights, or training data. Instead, they query the model with selected inputs and observe the corresponding outputs or predictions to infer the model parameters for replicating the functionality of a model. Since we focus on the link inference attacks toward the training graph, this line of attacks is not under consideration in the current scope.

Defense for Inference attacks. Some defense methods have been proposed to defend inference attacks and they can be roughly grouped into three categories: 1) by employing differential privacy [31], [4], [40], [48], [45], [26]; 2) by adding regularization terms in the loss function [18], [23], [29], [34]; 3) by adding noise in the model predictions [15]. The first two categories of solutions inevitably disrupt the training process of a target model, thus incurring considerable model accuracy degradation. For example, the L2-Regularizer [32] and Min-Max Game [23] added additional terms for overfitting prevention, inevitably degrading the accuracy of the target model. The differential privacy methods suffer considerable utility loss, as presented in the pioneering work [27], where a target GNN model has an over 20% degradation in model accuracy when defending

against the node inference attack. On the other hand, MemGuard [15] was proposed to defend against membership inference attacks, by adding noise vectors to the prediction vectors of a trained model for distortion, making it difficult for an attacker to infer if a queried data sample belongs to the member of the training dataset. The utility loss of the prediction vector is kept within bounds by moderating the intensity of the noise vectors. However, the protection of MemGuard for the GNN model is limited, as the noise is computed for distorting the relationship of the individual node prediction vector and its attributes, without taking the correlations among prediction vectors into consideration. Hence, it cannot capture the graph topology structure information when crafting the noises.

8. Conclusion

This paper has presented a novel defense solution for GNN models, called GRID, for countering link stealing attacks with the model utility guaranteed formally. Our GRID takes into account the graph topology to craft noise vectors for core nodes in the training graph to disguise the similarity of prediction vectors of adjacent nodes as those of n-hop indirectly connected neighboring nodes. We formulate the noise generation objective into an optimization problem and propose an algorithm for identifying a subset of nodes as core nodes for calculating noise vectors. The extensive experimental results on different datasets exhibit that GRID can effectively defend against different types of link-stealing attacks and outperform its counterparts in terms of defense performance and model utility tradeoffs.

Acknowledgments

This work was supported in part by NSF under Grants 2019511, 2348452, 2315613, 1937787, and 2325564. Any opinions and findings expressed in the paper are those of the authors and do not necessarily reflect the views of funding agencies.

References

[1] Graph convolutional networks. https://https://github.com/tkipf/gcn.

[2] Stealing links from graph neural networks. https://github.com/xinlei he/link_stealing_attack.

[3] Moco. https://github.com/PetarV-/GAT, 2017.

[4] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the ACM SIGSAC conference on computer and communications security (CCS)*, pages 308–318, 2016.

[5] Terry Caelli, Adnan Amin, Robert PW Duin, Mohamed Kamel, and Dick de Ridder. *Structural, Syntactic, and Statistical Pattern Recognition*. Springer, 2002.

[6] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *Proceedings of USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

[7] Varun Chandrasekaran, Kamalika Chaudhuri, Irene Giacomelli, Somesh Jha, and Songbai Yan. Exploring connections between active learning and model extraction. In *USENIX Security Symposium*, pages 1309–1326, 2020.

[8] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 1964–1974, 2021.

[9] Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

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

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

[12] 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.

[13] Seira Hidano, Takao Murakami, and Yusuke Kawamoto. Transmia: membership inference attacks using transfer shadow training. In *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, pages 1–10, 2021.

[14] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks. In *USENIX security symposium*, pages 1345–1362, 2020.

[15] 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 (CCS)*, pages 259–274, 2019.

[16] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. 2017.

[17] Klas Leino and Matt Fredrikson. Stolen memories: Leveraging model memorization for calibrated {White-Box} membership inference. In *USENIX Security Symposium (USENIX Security 20)*, pages 1605–1622, 2020.

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

[19] Zheng Li and Yang Zhang. Membership leakage in label-only exposures. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 880–895, 2021.

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

[21] Lingshuo Meng, Yijie Bai, Yanjiao Chen, Yutong Hu, Wenyuan Xu, and Haiqin Weng. Devil in disguise: Breaching graph neural networks privacy through infiltration. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1153–1167, 2023.

[22] 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 *Proceedings of IEEE symposium on security and privacy (S&P)*, pages 739–753.

[23] 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*, pages 634–646, 2018.

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

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

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

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

[28] Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. Updates-leak: Data set inference and reconstruction attacks in online learning. In *Proceedings of 29th {USENIX} Security Symposium*, pages 1291–1308, 2020.

[29] Ahmed Salem, Yang Zhang, Mathias Humbert, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *Network and Distributed Systems Security Symposium (NDSS)*, 2019.

[30] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[31] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the ACM SIGSAC conference on computer and communications security (CCS)*, pages 1310–1321, 2015.

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

[33] 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.

[34] Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of machine learning models. In *Proceedings of {USENIX} Security Symposium*, 2021.

[35] Liwei Song, Reza Shokri, and Prateek Mittal. Privacy risks of securing machine learning models against adversarial examples. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 241–257, 2019.

[36] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[37] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction {APIs}. In *USENIX security symposium*, pages 601–618, 2016.

[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[39] Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 36–52. IEEE, 2018.

[40] Di Wang, Minwei Ye, and Jinhui Xu. Differentially private empirical risk minimization revisited: Faster and more general. *Advances in Neural Information Processing Systems*, 30, 2017.

[41] Xiuling Wang and Wendy Hui Wang. Link membership inference attacks against unsupervised graph representation learning. In *Proceedings of the 39th Annual Computer Security Applications Conference*, pages 477–491, 2023.

[42] Max Welling and Thomas N Kipf. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[43] Wikipedia. Vertex cover. https://en.wikipedia.org/wiki/Vertex_cover.

[44] Bang Wu, Xiangwen Yang, Shirui Pan, and Xingliang Yuan. Adapting membership inference attacks to gnn for graph classification: Approaches and implications. In *IEEE International Conference on Data Mining (ICDM)*, pages 1421–1426, 2021.

[45] Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. Linkteller: Recovering private edges from graph neural networks via influence analysis. In *IEEE Symposium on Security and Privacy (SP)*, pages 2005–2024, 2022.

[46] Yixin Wu, Xinlei He, Pascal Berrang, Mathias Humbert, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. Link stealing attacks against inductive graph neural networks. *arXiv preprint arXiv:2405.05784*, 2024.

[47] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

[48] Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy, and Stacey Truex. Differentially private model publishing for deep learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 332–349. IEEE, 2019.

[49] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. *Advances in neural information processing systems*, 32, 2019.

[50] He Zhang, Bang Wu, Shuo Wang, Xiangwen Yang, Minhui Xue, Shirui Pan, and Xingliang Yuan. Demystifying uneven vulnerability of link stealing attacks against graph neural networks. In *International Conference on Machine Learning (ICML)*, pages 41737–41752, 2023.

Appendix A. Additional Calculation Details and Results

A.1. Similarity Calculations

$corr()$ and $cos()$ measure the Pearson correlation coefficient and cosine metrics between two vectors $\mathbf{v}_i$ and $\mathbf{v}_j$, which is calculated as

$$corr(\mathbf{X}, \mathbf{Y}) = \frac{\sum((x_i - \overline{X})(y_i - \overline{Y}))}{\sqrt{\sum(x_i - \overline{X})^2 \sum(y_i - \overline{Y})^2}}$$

$$cos(\mathbf{X}, \mathbf{Y}) = \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \cdot \sqrt{\sum_{i=1}^{n} y_i^2}}$$

where $\bar{X}$ and $\bar{Y}$ are the average values of $\mathbf{X}$ and of $\mathbf{Y}$.

A.2. Gradient Calculation

Here, the gradient $\nabla_s \mathcal{L}$ on the vector $\mathbf{s}$ is calculated on each entry. For $\lambda(v^a + s^a - v^* - s^*)$, we construct a vector $\mathbf{C}$, which is the same length as s, with elements defined as

- If corresponding to s^*, then $C_i = -1$.
- If corresponding to s^a, then $C_i = 1$.
- Otherwise, $C_i = 0$

Then, we have

$$\nabla_s[\lambda(v^a + s^a - v^* - s^*)] = \lambda \nabla_s(C^T s) = \lambda C \ . \quad (10)$$

TABLE 9. LINK STEALING ATTACKS-0, -1, AND -6 PERFORMANCE (%) ON GRAPHSAGE AND GIN UNDER FIVE DATASETS WITH AND WITHOUT OUR GRID. NOTE: 'A/B' ARE THE VALUES UNDER GRAPHSAGE AND GIN

		Citeseer		Cora		Pubmed		AIDS		ENZYMES	
		No Defense	GRID	No Defense	GRID	No Defense	GRID	No Defense	GRID	No Defense	GRID
Attack-0	Accuracy	85.8/85.5	65.1/64.7	84.4/84.1	62.2/61.9	77.6/77.3	60.6/60.4	74.0/73.9	59.4/57.1	73.1/72.6	**56.9/56.6**
	Precision	76.6/76.2	65.4/64.9	75.6/75.4	61.7/62.3	68.4/68.2	56.4/56.0	51.4/55.3	**50.2/51.3**	52.5/52.2	50.6/52.4
	Recall	97.1/96.7	63.7/63.4	94.9/94.6	62.5/62.0	93.5/93.2	62.1/61.7	96.7/96.2	**62.1/63.4**	97.7/97.4	62.5/62.0
	AUC	90.0/90.9	68.1/67.8	85.8/84.6	67.5/67.1	85.4/83.7	66.7/66.3	69.5/68.9	58.5/58.1	62.2/61.8	**55.2/55.0**
Attack-1	Accuracy	88.7/88.9	66.6/65.8	86.1/86.4	65.1/64.3	83.9/84.3	61.6/61.2	70.4/70.2	54.1/54.2	68.1/68.2	**52.3/52.4**
	Precision	85.9/86.1	66.0/66.3	84.1/84.3	65.7/65.5	76.5/76.4	61.3/60.2	71.3/71.0	54.3/54.0	67.1/67.3	**53.2/53.1**
	Recall	94.6/94.8	67.4/67.2	87.1/87.3	64.8/65.0	88.0/88.2	61.5/62.4	69.2/69.4	53.5/51.3	69.1/69.2	**51.8/52.0**
	AUC	91.3/95.5	72.3/72.4	87.9/93.0	71.8/71.9	87.3/87.5	69.6/69.4	71.7/71.6	**57.8/57.7**	73.3/73.2	59.0/56.1
Attack-6	Accuracy	91.4/90.9	67.2/66.9	92.2/89.7	67.6/67.1	90.2/89.7	68.0/66.4	93.4/92.8	67.5/63.4	81.6/81.2	**64.3/63.6**
	Precision	89.3/88.7	65.4/66.8	93.4/90.4	67.9/67.2	89.5/88.8	68.9/67.3	89.9/89.3	68.7/66.1	76.3/75.7	**64.6/64.0**
	Recall	92.5/92.0	68.3/67.0	92.0/89.6	68.5/66.1	91.6/91.0	67.8/66.2	97.8/97.2	66.9/62.3	87.9/87.2	**64.1/63.5**
	AUC	97.3/96.8	70.4/68.8	95.6/95.0	69.7/67.1	96.2/95.7	70.1/69.4	97.1/96.5	73.1/68.8	88.4/87.7	**67.4/66.9**
Model Accuracy		74.8/72.7	74.8/72.7	77.6/76.2	77.6/76.2	85.4/82.3	85.4/82.3	70.2/72.5	70.2/72.5	71.1/71.9	71.1/71.9

For $\mu \sum_a (s^a)$, we have

$$\nabla_s[\mu(\sum_a s^a)] = \mu \nabla_s(\mathbf{1}^T \mathbf{s}) = \mu \mathbf{1}^T \ , \tag{11}$$

where $\mathbf{1}$ is the vector of 1 as the same length as s. For $\nu(\|\mathbf{s}\| - \theta)$, we have

$$\nabla_s[\nu(\|\mathbf{s}\| - \theta)] = \nu \nabla_s \|\mathbf{s}\| = \nu \frac{s}{\|\mathbf{s}\|} \ . \tag{12}$$

Combine them together, we have:

$$\nabla_s \mathcal{L} = \nabla_s D_i + \lambda C + \mu \mathbf{1}^T + \nu \frac{s}{\|\mathbf{s}\|} \ . \tag{13}$$

A.3. Counterpart Defense Methods

L2-Regularizer. The first proposed membership inference attack [32] dealt with overfitting to lead to its success. It is also considered an elementary defense method by employing the L2-Regularizer to prevent overfitting. The main idea of L2-Regularizer is to add a penalty term, by calculating the sum of the L2-norm value of the weights in the target model, as the loss function.

Dropout. Dropout [36] is considered as another method to prevent the model overfitting, by randomly turning off some neurons (i.e., setting their outputs to 0) during each training iteration of the neural network. This way prevents the model from being sensitive to some particular neurons since any at-risk neuron is turned off at random to make its weight value as average as possible.

Min-Max Game. Inspired by the L2-Regularizer, authors in [23] proposed the Min-Max Game method, which added a special adversarial regularizer to defend against the membership inference attacks. This regularizer aims at maximizing membership privacy by narrowing the difference between the prediction vectors of the model on its training set and on other data samples with the same underlying distribution.

GAP. GAP was proposed in [26] as the latest and novel differentially private GNN based on aggregation perturbation rather than training graph perturbation. By adding stochastic noise to the aggregation function of a GNN, GAP statistically obfuscates the presence of a single edge (edge-level privacy) or a single node and all its adjacent edges (node-level privacy). GAP has a parameter privacy budget, i.e., ϵ, to tune the privacy-utility trade-off, where a lower privacy budget leads to stronger privacy guarantees but reduced utility.

MemGuard. MemGuard was proposed in [15] to defend the membership inference attack by randomly adding noise to the prediction vectors of a target ML model. The noise vector aims to counter random guessing results for the membership inference classifier by removing the membership pattern on the prediction vector that is related to the corresponding query sample. We employ the MemGuard to add noise vectors to the prediction vectors of all nodes in the training graph and control the distortion budget to evaluate its defense performance.

A.4. Defense Performance on GraphSAGE and GIN

The defense performance for attack-0, attack-1, attack-6 on GraphSAGE and GIN are shown in Table 9. Since our GRID is independent of specific GNN structure designs, it consistently performs well on GraphSage and GIN. For example, GRID decreases the accuracy(%) of the strongest Attack-6 to 67.2, 67.6, 68.0, 67.5, 64.3 and 66.9, 67.1, 66.4, 63.4, 63.6 for GraphSAGE and GIN on five datasets.

Appendix B.
Meta-Review

The following meta-review was prepared by the program committee for the 2025 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

B.1. Summary

The paper introduces the novel defense GRID against similarity-based link stealing attacks in Graph Neural Networks (GNNs). The idea of GRID is to introduce the noise to the prediction vectors of core nodes to ensure that the similarity of adjacent nodes in the graph is equivalent to n-hop neighbors while providing no loss in model accuracy. The proposed defense methodology was experimentally shown to effectively minimize attacks' success rate and outperform the existing methods w.r.t the utility-privacy tradeoff.

B.2. Scientific Contributions

- Addresses a Long-Known Issue
- Provides a Valuable Step Forward in an Established Field

B.3. Reasons for Acceptance

1) The paper addresses an important, long-known issue of protecting GNNs against link inference attacks in training graphs. The authors introduce a novel defensive mechanism that considers the graph's topology and the characteristics of the GNN aggregation mechanism.
2) The paper provides a valuable step forward in an established field. Unlike other defense models with utility-privacy tradeoffs, the authors provide the first defense with a formal guarantee of model utility. The paper introduces the idea of obfuscating prediction vectors of adjacent nodes with their n-hop neighbors and selecting the core nodes that cover all links instead of adding noise to the graph's nodes.
3) The evaluation considers various datasets, knowledge types of the attacker, target models, and defense results. Comparison experiments, ablation studies, and sensitivity analysis show promising results.