Learning-based Privacy-Preserving Graph Publishing Against Sensitive Link Inference Attacks

Yucheng Wu†, Yuncong Yang†, Xiao Han, Leye Wang, Junjie Wu

***Abstract*—Publishing graph data is widely desired to enable a variety of structural analyses and downstream tasks. However, it also potentially poses severe privacy leakage, as attackers may leverage the released graph data to launch attacks and precisely infer private information such as the existence of hidden sensitive links in the graph. Prior studies on privacy-preserving graph data publishing relied on heuristic graph modification strategies and it is difficult to determine the graph with the optimal privacy–utility trade-off for publishing. In contrast, we propose the first *privacy-preserving graph structure learning framework against sensitive link inference attacks*, named PPGSL, which can automatically learn a graph with the optimal privacy–utility trade-off. The PPGSL operates by first simulating a powerful surrogate attacker conducting sensitive link attacks on a given graph. It then trains a parameterized graph to defend against the simulated adversarial attacks while maintaining the favorable utility of the original graph. To learn the parameters of both parts of the PPGSL, we introduce a secure iterative training protocol. It can enhance privacy preservation and ensure stable convergence during the training process, as supported by the theoretical proof. Additionally, we incorporate multiple acceleration techniques to improve the efficiency of the PPGSL in handling large-scale graphs. The experimental results confirm that the PPGSL achieves state-of-the-art privacy–utility trade-off performance and effectively thwarts various sensitive link inference attacks.**

***Index Terms*—Link Inference Attack, Privacy Protection, Graph Publishing, Graph Neural Network, Graph Learning**

†*Equal contribution.*

Manuscript received 2 December 2024; revised 6 June 2025 and 7 July 2025; accepted 22 July 2025. Date of publication XX August 2025; date of current version XX August 2025. The authors thank the senior editor, associate editor, and anonymous reviewers for their guidance and constructive comments that have tremendously improved the paper. X. Han and J. Wu are supported in part by the National Key R&D Program of China (2023YFC3304700). X. Han is supported in part by the National Natural Science Foundation of China under grant No. 72071125 and 72031001. J. Wu is supported in part by the National Natural Science Foundation of China under grant No. 72031001, 72242101, and the Outstanding Young Scientist Program of Beijing Universities (JWZQ20240201002). *(Corresponding author: Xiao Han.)*

Yucheng Wu and Leye Wang are with the Key Lab of High Confidence Software Technologies, Peking University, Ministry of Education, Beijing 100871, China, and the School of Computer Science, Peking University, Beijing 100871, China (e-mail: wuyucheng@stu.pku.edu.cn; leyewang@pku.edu.cn).

Yuncong Yang is with Key Laboratory of Interdisciplinary Research of Computation and Economics (Shanghai University of Finance and Economics), Ministry of Education, Shanghai 200433, China, and the School of Information Management and Engineering, Shanghai University of Finance and Economics, Shanghai 200433, China (e-mail: yycphd@163.sufe.edu.cn).

Xiao Han and Junjie Wu are with the Key Laboratory of Data Intelligence and Management, Beihang University, Ministry of Industry and Information Technology, Beijing 100191, China, and the School of Economics and Management, Beihang University, Beijing 100191, China (e-mail: xh_bh@buaa.edu.cn; wujj@buaa.edu.cn)

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the author. The material includes proof of theoretical analysis and additional experimental results. Contact wuyucheng@stu.pku.edu.cn for further questions about this work.

I. Introduction

GRAPH data are ubiquitous in our daily lives, spanning the realms of social relationships [1], [2], communication networks [3], and traffic networks [4], *etc*. Owing to the abundant information in graph data, it is common practice for data holders to publish them for academic and economic benefits. For example, universities and research institutes such as SNAP [5] and AMiner [6] collect and release substantial volumes of graph data, significantly fostering the development of graph data mining; social media share their data via open APIs (*e.g.*, Facebook [7]) for business. However, sharing graph data without adequate protection may result in severe privacy leakage problems, especially when encountering various inference attacks [8]–[10]. According to privacy laws, including GDPR [11], it is imperative to protect private information so that users are unwilling to be exposed to data publications [12]. This leads to an urgent need for privacy protection measures when sharing graph data.

This work focuses on one of the most common inference attacks on graphs: *sensitive link inference attacks* [13]–[15]. These attacks use various techniques on released graph data to accurately deduce hidden sensitive links between users, posing significant risks to their private information. Hidden sensitive links, such as private friendships and confidential transactions, are the links that users intentionally conceal to protect their privacy and remain invisible to the public. For example, the Facebook platform allows its users to make their partial friendships private and invisible from others, and these hidden friendships are regarded as sensitive links. Despite the invisibility of sensitive links in the published graph, they can still be inferred because of the pronounced similarity between their connected node pairs (*i.e.*, sensitive node pairs). Preliminary empirical analyses show that node pairs with hidden links exhibit significantly greater similarity across various metrics than do unlinked node pairs in Table I. This suggests that attackers can easily infer the presence of hidden links by evaluating the similarity between unconnected node pairs.

To defend against sensitive link inference attacks, a straightforward strategy is to reduce the structural proximity of sensitive node pairs, which applies to both attributed and unattributed graphs. However, this strategy unavoidably introduces disturbance to the original graph, thereby leading to a degradation of graph data utility. Therefore, researchers strive to improve the protection of sensitive link privacy while

TABLE I
STRUCTURAL PROXIMITY (INCLUDING AVERAGE SHORTEST PATH LENGTH AND DISCONNECTION RATIO), ATTRIBUTE COSINE SIMILARITY AND EMBEDDING COSINE SIMILARITY (WHERE THE EMBEDDING IS PRODUCED BY GRAPHSAGE [16]) OF DIFFERENT TYPES OF NODE PAIRS

Dataset	Node pair type	Avg. shortest path length	Discon. ratio	Attribute similarity	Embedding similarity
Cora	w/ visible link	1.000	0.000	0.011	0.900
	w/ hidden link	3.206	0.164	0.011	0.702
	w/o link	7.007	0.254	0.003	0.499
LastFMAsia	w/ visible link	1.000	0.000	0.010	0.978
	w/ hidden link	2.480	0.089	0.010	0.947
	w/o link	5.560	0.124	0.008	0.820

minimizing the loss of graph utility, *i.e.*, seeking a graph that strikes an optimal privacy–utility trade-off for publishing. Concerning the computational infeasibility of enumerating all potential graph structures, they resort to heuristic graph modification solutions. More specifically, they often randomly select a small set of node pairs from the original graph and greedily identify the best link modifications (*e.g.*, adding or deleting links) among these candidate node pairs for a favorable privacy–utility trade-off [13], [15]. Nevertheless, the main limitation of these heuristic solutions is the lack of guarantees for reaching the optimal privacy–utility trade-off, which makes them prone to becoming stuck in local optima. To overcome the limitations of existing methods, we aim to develop a solution that can derive an optimal graph structure for release within a reasonable computational time.

Recent advances in graph structure learning (GSL) have achieved remarkable success in solving graph-related tasks [17]. GSL treats the adjacency matrix of a graph as a set of continuous parameters, enabling the application of optimization techniques to refine the parameterized matrix and determine the optimal graph structure for specific objectives. However, most existing GSL methods focus on designing objective functions that enhance robustness, smoothness, and task performance [18], [19], with limited attention given to privacy protection. **This work proposes and investigates the problem of privacy-preserving graph structure learning, aiming to identify the optimal graph structure that achieves the best privacy–utility trade-off for graph publishing**. To address this problem, we face the following challenges:

Challenge 1: Differentiable and universal privacy protection objective. It is crucial to design an appropriate objective to navigate the graph learning process toward protecting sensitive link privacy. While existing privacy-related studies have proposed some privacy protection objectives, they are typically nondifferentiable and thus unsuitable for graph structure learning [15]. How can we create a differentiable privacy objective that provides a supervisory signal throughout the learning process? In addition, how can we guarantee that the learned graph, guided by this objective, is robust enough to withstand a variety of sensitive link inference attacks?

Challenge 2: Optimal privacy–utility trade-off. Solely focusing on privacy protection can severely degrade the utility of the learned graph, thereby rendering graph-based tasks ineffective. How can we design a utility objective function that ensures that the learned graph retains as much utility as possible from the original graph? Furthermore, how can we balance privacy and utility to jointly learn a graph that maximally preserves privacy while minimizing utility loss?

Challenge 3: Effective and efficient training protocol. The graph learning process is complex and involves multiple objectives and many trainable parameters (especially in large graphs). While most GSL methods adopt an alternating or end-to-end training protocol for the components corresponding to different objectives [17], these protocols may expose privacy during training and lack guarantees of stable convergence. How can we design an effective and efficient training protocol to avoid suboptimal solutions and unstable convergence while ensuring high efficiency and scalability?

By jointly considering the above challenges, this work makes the following contributions:

- To the best of our knowledge, we are the first to formalize the problem of learning a privacy-preserving graph against sensitive link inference attacks, which aims to learn graphs for publishing with the optimal privacy–utility trade-off. Our research broadens both the realms of privacy-preserving and graph structure learning studies.
- We propose a generic **P**rivacy-**P**reserving **G**raph **S**tructure **L**earning (*i.e.*, **PPGSL**) framework, which is designed to automatically learn an optimal graph structure that protects sensitive link privacy. Within the PPGSL, we develop an effective and differentiable privacy protection objective derived by establishing a surrogate attack model and conducting inference on sensitive links (tackling Challenge 1). We also devise a utility objective that minimizes the distortion between the learned and original graphs, guiding the learning process toward an optimal privacy–utility balance alongside the privacy protection objective (tackling Challenge 2). To ensure stable convergence, we design a secure iterative training protocol (*i.e.*, SITP) and introduce speed-up strategies to further increase scalability and efficiency (tackling Challenge 3).
- Rigorous theoretical analyses confirm the key benefits of PPGSL with SITP, including stable convergence, optimal privacy–utility trade-off, and universal defense capability.

We conduct extensive empirical evaluations on six real-world graph datasets across two common utility tasks (node classification and link prediction). Our results show that the PPGSL achieves a superior privacy–utility trade-off compared with baseline methods and effectively defends against various sensitive link inference attacks.[1]

II. PROBLEM FORMULATION

A. Preliminaries

Let $G = (V, E) = (A, X)$ be a graph, where V is the set of nodes and $E \subseteq V \times V$ represents the set of publicly observable links on the graph. $N = |V|$ is the total number of nodes. For the unweighted graph, we use $A = \{w_{ij}\} \in \{0, 1\}^{N \times N}$ to denote the adjacency matrix of G, where the entry $w_{ij} = 1$ if the link $\langle v_i, v_j \rangle \in E$; otherwise, $w_{ij} = 0$. For the weighted

[1]The code is available at https://github.com/wuyucheng2002/PPGSL, along with detailed parameter settings and experimental results of the PPGSL.

graph, $A = \{w_{ij}\} \in \mathbb{R}^{N\times N}$, where w_{ij} is the edge weight of $\langle v_i, v_j\rangle$. $X \in \mathbb{R}^{N\times D}$ denotes the node feature matrix, where D is the dimension of the node features. Let $G' = (V, E') = (A', X')$ denote the released graph data[2].

Definition 1 (Sensitive links)**.** *The set of sensitive links, denoted as $E_s \subseteq V \times V$, exists yet remains unobservable on graph G due to privacy concerns, signifying that $E_s \cap E = \varnothing$.*

Definition 2 (Sensitive node pair)**.** *A pair of nodes v_i and v_j is called a sensitive node pair if $\langle v_i, v_j\rangle \in E_s$.*

Furthermore, we define *nonexistent links* as pairs of nodes that are not connected by any link (neither a publicly observable nor sensitive link) in the original graph, denoted by E_n. Thus, we have $E_n \cap E_s = \varnothing$ and $E_n \cap E = \varnothing$.

B. Sensitive Link Inference Attacks

Attackers' goal. Attackers aim to accurately infer the existence of sensitive links E_s within the graph G. Taking Facebook as an example, it shares certain user attributes (*e.g.*, nickname and age) and public friendship data via an open API, typically for commercial purposes. However, this data sharing also poses a threat to the privacy of users who choose to hide their private friendships from a public view. Specifically, malicious data users (*i.e.*, attackers) can retrieve the public social network data via the API and leverage it to conduct inference attacks on the hidden private friendships between the privacy-sensitive users. Leaks of private friendships not only violate user privacy but can also have more severe consequences. For example, attackers may exploit hidden links to defraud or blackmail users or execute advanced network attacks such as Phishing attacks [20] and Sybil attacks [21].

Attackers' knowledge. Attackers can access the released graph G' on public platforms but do not have any information on the sensitive links E_s.

Attack model. Attackers can employ various algorithms to establish an attack model (denoted as $\mathcal{M}$), such as employing network embedding similarity to infer link existence [22]. Note that the true sensitive links are unavailable to attackers, and the only available information is G'. Therefore, they may sample a set of existing/nonexistent links from G' as positive/negative samples to construct a dataset used to train the attack model $\mathcal{M}$ and then conduct inference on E_s via the well-trained model. This is reasonable because sensitive pairs are expected to exhibit high similarity, such as connected node pairs (as demonstrated in Sec. I). Formally, given G' and the chosen attack model $\mathcal{M}$, the training process aims to minimize the expected inference error on E'_p and E'_n, where $E'_p \subset E'$ denotes the existing links sampled from G' and E'_n represents the nonexistent links sampled from G':

$$\mathcal{M}^* = \arg\min_{\mathcal{M}}(error(\mathcal{M}(G'), E'_p \cup E'_n)) \tag{1}$$

After the training process of the attack model, attackers utilize the well-trained model $\mathcal{M}^*$ to infer the existence of E_s.

[2]Note that we do not add or remove nodes on the graph G (*i.e.*, $V' = V$), so we omit V' here.

C. Problem Definition

Now, we define our **privacy-preserving graph structure learning problem** as follows: Given an original graph G and a set of sensitive links E_s, our objective is to learn a graph G' via a graph structure learning function $\mathcal{H}_\theta(\cdot)$ with a set of parameters θ such that $G' = \mathcal{H}_\theta(G)$. The learned graph G' should maximize the error of the well-trained attack model $\mathcal{M}^*$ in inferring the sensitive links E_s to protect privacy; G' should also minimize the data distortion for utility preservation. Mathematically, our problem can be formulated as follows:

$$\textbf{Privacy goal: } \max_{\theta}(error(\mathcal{M}^*(\mathcal{H}_\theta(G)), E_s)), \tag{2}$$

$$\mathcal{M}^* = \arg\min_{\mathcal{M}}(error(\mathcal{M}(\mathcal{H}_\theta(G)), E'_p \cup E'_n)) \tag{3}$$

$$\textbf{Utility goal: } \min_{\theta}(dist(\mathcal{H}_\theta(G), G)) \tag{4}$$

where $dist(\cdot,\cdot)$ is a general distance function used to quantify the difference between the topologies of two graphs.

III. Proposed Framework: PPGSL

In this section, we elucidate our proposed privacy-preserving graph structure learning (*i.e.*, PPGSL) framework, which learns a graph with the optimal privacy–utility trade-off against sensitive link inference attacks. In particular, the PPGSL consists of two iterative training modules (*i.e.*, *surrogate attack module* and *privacy-preserving graph learner module*), and its overview is depicted in Fig. 1. The *surrogate attack module* imitates an attacker conducting inference attacks in two phases. First, it constructs a surrogate attack model that is based on a given learned graph; second, it uses the attack model to infer the existence of sensitive links and induces a privacy leakage risk due to the current graph. The privacy leakage information is subsequently delivered to the other graph learner module as a supervisory signal of the learning process. In addition, the *privacy-preserving graph learner module* parameterizes the graph structure to be trainable and manages the learning process. It updates the graph structure to reduce privacy leakage while controlling the distortion to maintain data utility. We also propose a *secure iterative training protocol* to iteratively train the two modules, enhancing both the privacy level of the learned graph and the stability of the optimization process. In addition, several *speed-up strategies* are put forth to improve the scalability and efficiency of the PPGSL.

A. Surrogate Attack Module

To guide our graph learner in producing a privacy-preserving graph, we need to provide supervisory signals that direct the learned graph structure toward minimizing privacy leakage. To achieve this goal, we first establish a surrogate attack model to simulate inference attacks on sensitive links. The privacy leakage risk induced by this surrogate attack model then serves as a supervisory signal, instructing the graph learner to produce a graph that resists inference attacks.

Previous studies have proposed the use of heuristic metric-based surrogate attack models, such as the resource allocation

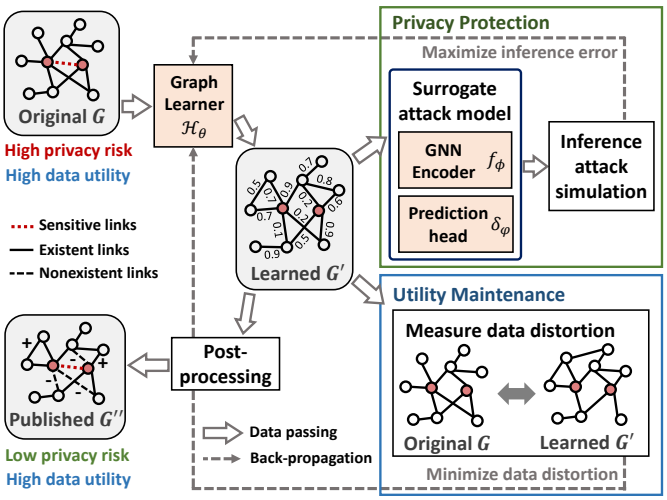


Fig. 1. Overview of the PPGSL.

index [15], for similar purposes. However, these approaches face two significant limitations. First, simple heuristics capture limited information, providing only local structure proximity details, which may result in weak surrogate attack models and imprecise supervisory signals. Second, existing heuristic metric-based surrogate attack models are nondifferentiable, making them unsuitable for providing signals that support gradient back-propagation in our learning-based graph structure optimization process.

To address these issues, we propose a machine learning (ML)-based surrogate attack model. As described in Sec. I, the relatively high structural proximity between sensitive node pairs significantly contributes to the ease with which attackers infer sensitive links. By leveraging ML techniques, we develop a surrogate attack model that effectively exploits structural proximity information within the graph. Furthermore, this ML-based surrogate attack model ensures that both the training and inference processes are fully differentiable.

1) Surrogate Attack Model Architecture: Our surrogate attack model comprises a graph neural network (GNN) encoder and a prediction head. Its forward propagation is differentiable, allowing us to utilize its information to direct the optimization of graph structure learning through gradient back-propagation. Moreover, GNN encoders are widely used by attackers and present effective attack performance in previous works [14], making our surrogate attack model highly representative.

In particular, we employ a GNN encoder (denoted as f_ϕ) to generate informative node embeddings, as GNNs are highly expressive for representing graph data [23]. As analyzed in Sec. I, node pairs with sensitive links exhibit increased structural proximity, a vulnerability often exploited by attackers. The GNN effectively identifies this relationship—nodes with close structural distance will produce embeddings with high similarity as the GNN iteratively aggregates neighborhood information. Generally, any GNN model, such as GCN [24] or GraphSAGE [16], can serve as the encoder.

Additionally, the prediction head (denoted as δ_φ) determines the existence of a link on the basis of the learned node embeddings of its two ends. It uses a similarity metric such as cosine similarity or inner product [14] or employs multilayer perceptrons (MLPs) that concatenate the embeddings of the target link's two ends as input vectors and output the similarity score of the link. The prediction head further includes a sigmoid function $\sigma(x) = 1/(1+e^{-x}) \in (0,1)$ to convert the similarity score into the predicted probability of a link's existence. Specifically, a probability near 0 suggests dissimilar node embeddings and the absence of the target link, whereas a probability close to 1 indicates similar node embeddings and the presence of the target link. The GNN encoder f_ϕ and prediction head δ_φ together form an effective and representative attack method, which substitutes $\mathcal{M}$ in Eq. 2.

2) Surrogate Attack Model Training Objective: Essentially, our surrogate attack model is trained to infer the existence of links on the basis of node embedding similarity in the learned graph. Given a learned graph $G' = \mathcal{H}_\theta(G)$, we first construct the training dataset for the surrogate attack model by sampling an edge set E'_p from G' as the positive samples and a set of nonexistent edges E'_n as the negative samples. Then, we train the surrogate attack model by making the predicted link existence probability close to the edge weight for positive samples and close to 0 for negative samples. The objective is as follows:

$$\begin{aligned}\min_{\phi,\varphi} \mathcal{L}_{attack} =& \mathbb{E}_{\langle v_i,v_j\rangle\in E'_p} CE\left(\delta_\varphi(z'_i,z'_j), w'_{ij}\right) \\ &+ \mathbb{E}_{\langle v_i,v_j\rangle\in E'_n} CE\left(\delta_\varphi(z'_i,z'_j), 0\right)\end{aligned} \tag{5}$$

where $CE(\cdot,\cdot)$ is the cross-entropy function. We use $Z' = f_\phi(G')$ to denote the learned node representations on graph G', z'_i denotes the representation of node v_i, and w'_{ij} is the weight of edge $\langle v_i, v_j\rangle$. Intuitively, using Eq. 5, connected nodes are forced to have similar embeddings (the degree of similarity relates to the edge weight), whereas unconnected weights are repelled to exhibit orthometric embeddings.

3) Inference Attack Simulation: Once the training of the surrogate attack model is complete, it can be used to infer the existence of sensitive links. By inputting each sensitive link into the model, we obtain a predicted probability indicating its existence. Essentially, the simulation results provide an estimate of the privacy leakage of sensitive links. To precisely estimate the risk, we ensure that the surrogate attack model is adequately trained for convergence.

B. Privacy-Preserving Graph Learner Module

The privacy-preserving graph learner is a nonlinear learning model that transforms an original graph into a new privacy-preserving graph. It comprises a graph model architecture and learning objectives that address both privacy protection and utility maintenance goals. We will subsequently elaborate on these components.

1) Graph Model Architecture: Like various graph model architectures in recent studies for other graph learning purposes, such as graph denoising [25], [26], we apply a straightforward full graph parameterization (FGP) approach to construct the graph model architecture for our privacy-preserving aim. FGP treats each entry of the adjacency matrix of a graph as an independent parameter and allows the learning of any adjacency matrix. We use $\mathcal{H}$ with parameters θ to denote

the graph model architecture using FGP, *i.e.*, $G' = \mathcal{H}_\theta(G)$. Since θ may encompass nonsymmetrical and negative values, which are impermissible for an adjacency matrix, we refine θ to derive the learned adjacency matrix with symmetrization and truncation techniques. In particular, symmetrization is executed as $\theta^s = (\theta + \theta^\top)/2$. Additionally, we confine the values of θ^s_{ij} between 0 and 1, truncating those falling below 0 or exceeding 1, $\theta^{st}_{ij} = \min\{\max\{\theta^s_{ij}, 0\}, 1\} \in [0, 1]$. In summary, $\mathcal{H}_\theta(G)$ can be determined as follows:

$$\mathcal{H}_\theta(G) = \min\left\{\max\left\{\frac{\theta + \theta^\top}{2}, 0\right\}, 1\right\} \tag{6}$$

As a result, the adjacency matrix $A' = \mathcal{H}_\theta(G) \in [0, 1]^{N\times N}$ is continuous,[3] and we can interpret its term w'_{ij} as the weight of edge $\langle v_i, v_j\rangle$ throughout the training phase. Notably, while we use FGP here for its simplicity and flexibility, the proposed PPGSL framework can accommodate alternative graph model architectures in place of FGP.

2) Privacy Protection Objective: As described in Sec. III-A, given a graph G' produced by graph learner, the surrogate attack module trains an attack model and simulates inference attacks on sensitive links. The results of the attack serve as an estimate of the privacy leakage of sensitive links in the produced graph. To guide the graph learner toward reducing privacy leakage, we use the inference results of the surrogate attack model on the current learned graph as a supervisory signal. Specifically, we aim to update the graph structure so that the surrogate attack model is more likely to misclassify sensitive links as nonexistent, thereby achieving the privacy protection goal. This objective is formulated as:

$$\min_\theta \mathcal{L}_{priv} = \mathbb{E}_{\langle v_i, v_j\rangle \in E_s} CE\left(\delta_\varphi(z'_i, z'_j),\, 0\right) \tag{7}$$

3) Utility Maintenance Objective: It is also important to maintain the utility level of graph data for downstream applications. As delineated in Eq. 4, we measure the loss of data utility by the distortion between the original graph and the learned graph. The original graph data are considered to possess the highest level of data utility, and any data distortion will lead to a deterioration in utility, as noted in prior studies [27]–[29]. In essence, more pronounced data distortion indicates greater loss in data utility. Specifically, the data distortion can be ascertained via distance metrics, *e.g.*, the Frobenius norm. We aspire to guide the learned graph structure toward minimal distortion; hence, the utility maintenance objective function can be instantiated as follows:

$$\min_\theta \mathcal{L}_{util} = \|A - A'\|_F^2 \tag{8}$$

Optimizing Eq. 8 guarantees that the learned adjacency matrix A' aligns closely with the original adjacency matrix A, thereby preserving the utility of the learned graph data. Since this objective provides a universal gauge and performs well across various downstream tasks, it is particularly suitable for graph publishing scenarios, where downstream tasks are unknown.

[3]Since we only learn the graph structure and do not generate new node attributes, *i.e.*, $G' = (A', X)$, we also use the expression $A' = \mathcal{H}_\theta(G)$ for brevity.

4) Overall Objective: To derive the overall objective for optimizing the graph learner, the privacy protection and utility maintenance objectives can be combined as follows:

$$\min_\theta \mathcal{L}_{learner} = \mathcal{L}_{priv} + \alpha\mathcal{L}_{util} \tag{9}$$

where $\alpha \in [0, +\infty)$ functions as a hyperparameter to control the trade-off between the privacy protection effect and the data utility level. A smaller α indicates intensified privacy protection, albeit at the expense of reduced data utility.

5) Postprocessing: If the original graph G is unweighted, we need to recover the learned adjacency matrix $A' = \mathcal{H}_\theta(G)$ with continuous values into an unweighted adjacency matrix $\tilde{A}$ with *discrete* values for publishing. To this end, we employ a postprocessing method using an independent Bernoulli distribution $\mathcal{T}$ with the edge weight A'_{ij} as the probability parameter to discretize the learned adjacency matrix, *i.e.*, $A''_{ij} = \mathcal{T}(A'_{ij})$. Thus, the final published adjacency matrix A'' possesses entries of either 0 or 1.

C. Secure Iterative Training Protocol

Analytically, it is challenging to simultaneously optimize the two parameter groups in our framework's surrogate attack model and privacy-preserving graph learner. To address this, we introduce an innovative secure iterative training protocol (denoted as SITP) that iteratively trains both models. Unlike typical adversarial regularization approaches (*e.g.*, AdvReg [30]), which alternate training between the two models until they converge simultaneously, SITP focuses on training the privacy-preserving graph learner by using a well-trained surrogate attack model at each step of the gradient descent process to update the graph learner's parameters θ. In other words, SITP iteratively retrains a surrogate attack model and performs a gradient descent update for graph structure learning until the privacy-preserving graph learner converges. Formally, considering that the two sets of variables ϕ (of f_ϕ) and θ (of $\mathcal{H}_\theta$) in the PPGSL[4], SITP performs the following optimization procedure in the t-th iteration:

1) Reinitialize and optimize ϕ: At each iteration, reinitialize ϕ randomly and optimize it to minimize $\mathcal{L}_{attack}$ with θ fixed, yielding

$$\phi^{(t+1)} = \arg\min_\phi \mathcal{L}_{attack}(\phi, \theta^{(t)}) \tag{10}$$

2) Update θ: Fix $\phi^{(t+1)}$ and perform one step of gradient descent on θ to minimize $\mathcal{L}_{learner}(\phi^{(t+1)}, \theta)$, yielding

$$\theta^{(t+1)} = \theta^{(t)} - \eta\nabla_\theta\mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t)}) \tag{11}$$

where η is the learning rate.

Our proposed SITP offers several advantages in the privacy-preserving data publishing scenario:

- **Enhanced privacy preservation.** In the initial training phase, the surrogate attack model inevitably encodes the privacy information from the original graph, as it is trained to infer sensitive links on the basis of this information.

[4]φ (of δ_φ) can be seen as part of ϕ (of f_ϕ), as φ and ϕ are updated jointly.

If the attack model is continually trained without reinitialization, this privacy-related prior cannot be eliminated, leading to a significant divergence from the true attack model. Consequently, the graph learner may stray from developing a genuinely privacy-preserving graph. Reinitializing the attack model periodically in SITP, however, helps ensure that the learned graph achieves greater privacy preservation.

- **Stable convergence.** The surrogate attack model is optimized to converge in each update step, allowing the graph learner to converge more reliably. Theorem 1 validates the convergence property of the graph learner $\mathcal{H}_\theta$ under our proposed SITP.
- **Efficient training process.** Compared with training the graph learner, training the surrogate attack model is relatively straightforward and converges quickly because of its fewer parameters and simpler training objectives. Therefore, retraining the attack model after each gradient descent step of the graph learner incurs tolerable computational overhead and can even reduce the overall training time compared with typical training protocols.

We also empirically compare SITP with typical training protocols in terms of privacy–utility trade-offs, convergence, and training efficiency, as shown in Sec. V-B6.

D. Speed-up Strategies

The scalability and efficiency of the vanilla PPGSL may encounter limitations for two reasons:

(1) The space complexity of the graph learner $\mathcal{H}_\theta$, exacerbated by the FGP method, necessitates $\mathcal{O}(N^2)$ parameters for optimization, thereby posing a significant scalability issue when the graph is large.

(2) Each iteration for updating the learned graph G' requires retraining the surrogate attack model until convergence to achieve accurate privacy leakage estimation, which is time-consuming. Thus, we propose two speed-up strategies to increase the scalability and efficiency of the PPGSL as follows.

PPGSL-sparse. Since graphs are typically sparse in practice (*i.e.*, $|E| \ll N^2$), we can parameterize only the existing links and portions of nonexistent links in the original graph rather than parameterizing the whole adjacency matrix via the FGP method. Specifically, we define a parameter of the *sampling factor* denoted by k, and we randomly sample a set of $(k\times|E|)$ nonexistent links. We then parameterize the edge weight of each existent and sampled nonexistent link. By this strategy, we can substantially reduce the computational complexity of the PPGSL from $\mathcal{O}(N^2)$ to $\mathcal{O}((k+1)\times|E|)$. Moreover, a larger sampling size k for nonexistent links leads to enhanced privacy protection (owing to the potential structural noise of adding edges) but with increased computational overhead.

PPGSL-skip. Considering the subtle variation in the learned graph structure within each update, we can persistently utilize the surrogate attack model derived from the preceding graph structure, thereby reducing the update frequency of the surrogate attack model and reducing the overall model training time. We define a parameter of *update interval*, symbolized by μ. Specifically, we retrain the surrogate attack model every μ iterations, as opposed to constant updates at each iteration. Amplifying μ may curtail time expenditures but yield less precise results.

The pseudocode of our proposed PPGSL framework with two speed-up strategies is shown in Algorithm 1.

Algorithm 1 Pseudocode of the PPGSL Framework

Input: the original graph $G=(A,X)$, the set of sensitive links E_s, hyper-parameters α, k, μ, the training epoch N_1 and learning rate η_1 of the surrogate attack model, the training epoch N_2 and learning rate η_2 of the privacy-preserving graph learner
Output: the learned graph structure A'' for publishing

Sample $(k\times|E|)$ nonexistent edges and construct the graph learner $\mathcal{H}_\theta$
Initialize parameters of the graph learner $\mathcal{H}_\theta$
Construct the surrogate attack model f_ϕ, δ_φ
for $e_2=0$; $e_2<N_2$; e_2++ **do**
  Generate a graph structure, $A' \leftarrow \mathcal{H}_\theta(G)$
  if $e_2 \,\%\, \mu = 0$ **then**
    Initialize parameters of surrogate attack model f_ϕ, δ_φ
    for $e_1=0$; $e_1<N_1$; e_1++ **do**
      Sample sets of existent and nonexistent links from A'
      Calculate $\mathcal{L}_{attack}$ in Eq. 5 based on A'
      Update parameters, $\phi \leftarrow \phi - \eta_1 \cdot \frac{\partial \mathcal{L}_{attack}}{\partial \phi}$, $\varphi \leftarrow \varphi - \eta_1 \cdot \frac{\partial \mathcal{L}_{attack}}{\partial \varphi}$
    end for
  end if
  Calculate $\mathcal{L}_{learner}$ in Eq. 9 based on f_ϕ, δ_φ, A', and A
  Update parameters, $\theta \leftarrow \theta - \eta_2 \cdot \frac{\partial \mathcal{L}_{learner}}{\partial \theta}$
end for
Generate a graph structure, $A' \leftarrow \mathcal{H}_\theta(G)$
Discretize the graph structure using Bernoulli sampling function $\mathcal{T}$, $A'' \leftarrow \mathcal{T}(A')$
return A''

IV. Theoretical Analyses

Rigorous theoretical analyses prove the convergence property of the PPGSL's training protocol, the optimality of its privacy–utility trade-off, and its generalizability across various attack methods. The detailed proofs are provided in Appendix A.

Theorem 1 (**Convergence of the PPGSL**). *In PPGSL training procedures, graph learner $\mathcal{H}_\theta$ converges under SITP.*

Proof Sketch. Under the training of the PPGSL with SITP, the following inequality holds from the t-th iteration to the $(t+1)$-th iteration (the detailed proof is in Proposition 1):

$$\mathbb{E}[\mathcal{L}_{learner}(\phi^{(t+1)},\theta^{(t+1)})] \leq \mathbb{E}[\mathcal{L}_{learner}(\phi^{(t)},\theta^{(t)})] \quad (12)$$

which implies that each full iteration (reinitializing and optimizing ϕ, then updating θ) results in a nonincreasing expected value of the loss function $\mathcal{L}_{learner}$. Since the expected value of $\mathcal{L}_{learner}$ is nonincreasing at each iteration and is assumed to be lower-bounded, by the Monotone Convergence Theorem, $\mathbb{E}[\mathcal{L}_{learner}(\phi,\theta)]$ converges to a stable value as the iteration index t increases. Therefore, graph learner $\mathcal{H}_\theta$ converges. □

Theorem 2 (**Empirical Optimal Privacy–utility Trade-off of the PPGSL**). *Minimizing the training objective $\mathcal{L}_{learner}$*

TABLE II
STATISTICS OF THE DATASETS.

Dataset	#Nodes	#Links	#Features	#Labels
PolBlogs	1,490	19,025	0	2
LastFMAsia	7,624	27,806	128	18
DeezerEurope	28,281	185,504	128	2
Cora	2,708	10,556	1,433	7
CiteSeer	3,327	9,104	3,703	6
PubMed	19,717	88,648	500	3

of the PPGSL achieves the empirical optimal privacy–utility trade-off at a specified utility level.

Proof Sketch. Relying on the Lagrangian dual method, we can prove that minimizing $\mathcal{L}_{learner}$ can be reinterpreted as minimizing privacy loss while adhering to a given utility constraint (the detailed proof is in Proposition 2). This ensures that the empirical optimal privacy–utility trade-off is attained. □

Theorem 3 (Generalized Privacy Protection Performance of the PPGSL). *The PPGSL provides a lower bound of the privacy protection level on sensitive links in a graph regardless of the sensitive link inference model adopted by attackers. The inference error probability $p(E_s \neq \mathcal{M}^*(G'))$ of any inference model $\mathcal{M}^*$ that attempts to infer E_s from the published graph G' is lower-bounded by:*

$$p(E_s \neq \mathcal{M}^*(G')) \geq \frac{H(E_s) - I(G'; E_s) - 1}{\log |\mathcal{E}_s|} \tag{13}$$

where $|\mathcal{E}_s|$ denotes the cardinality of the set of possible values of E_s, $I(\cdot,\cdot)$ represents mutual information, and $H(\cdot)$ denotes information entropy. Furthermore, this lower bound increases during the PPGSL training process.

Proof Sketch. Motivated by prior studies that confirmed a relationship between the inference success of any algorithm and mutual information measures [31], [32], we establish a connection between PPGSL and mutual information. Let $I(\cdot,\cdot)$ denote mutual information; then, the privacy goal of our problem can be reformulated as $\min_\theta I(G'; E_s)$. We aim for the learned graph G' to contain as little mutual information about sensitive links as possible. In the PPGSL training process, the mutual information $I(G'; E_s)$ decreases (the detailed proof is in Proposition 3).

According to Fano's inequality, the inference error probability $p(E_s \neq \mathcal{M}^*(G'))$ of any inference model $\mathcal{M}^*$ that attempts to infer E_s from G' is lower-bounded by $\frac{H(E_s)-I(G';E_s)-1}{\log |\mathcal{E}_s|}$, where $H(E_s)$ and $\log |\mathcal{E}_s|$ are constants for a given private information variable E_s. Consequently, as the mutual information $I(G'; E_s)$ decreases, the lower bound on the inference error probability $p(E_s \neq \mathcal{M}^*(G'))$ increases, where $\mathcal{M}^*$ is any sensitive link inference model trained on the published graph G'. Therefore, the PPGSL inherently increases the inference error probability by maximizing this lower bound, regardless of the inference algorithm used. □

TABLE III
TRADITIONAL NODE PROXIMITY METRICS FOR LINK INFERENCE ATTACKS. $\mathcal{N}(x)$: THE NEIGHBOR SET OF VERTEX x; *Order*: THE MAXIMUM HOP OF NEIGHBORS NEEDED TO CALCULATE THE PROXIMITY SCORE.

Metrics	Formula	Order
Common Neighbor (CN)	$\lvert\mathcal{N}(x) \cap \mathcal{N}(y)\rvert$	first
Adamic-Adar (AA)	$\sum_{z \in \mathcal{N}(x) \cap \mathcal{N}(y)} \frac{1}{\log \lvert\mathcal{N}(z)\rvert}$	second
Resource Allocation (RA)	$\sum_{z \in \mathcal{N}(x) \cap \mathcal{N}(y)} \frac{1}{\lvert\mathcal{N}(z)\rvert}$	second

V. EXPERIMENTS

A. Experimental Setup

1) Datasets: We conduct experiments on six commonly used real-life datasets[5], including three social networks: *PolBlogs*, *LastFMAsia* and *DeezerEurope*; and three citation graphs: *Cora*, *CiteSeer* and *PubMed*. Their statistics are summarized in Table II. The dataset descriptions are as follows:

- **PolBlogs** [33] is a relationship network of political blogs, where nodes represent blogs and edges signify links extracted from their front pages. Each node is labeled as either liberal or conservative.
- **LastFMAsia** [34] is a social network of users from Asian countries on the music service LastFM. Nodes represent users, edges denote friendships, and node labels are users' home countries, with features based on preferred artists.
- **DeezerEurope** [34] is a social network of European Deezer users. Nodes represent users, and links represent mutual follower relationships. Node labels stand for gender, and features are derived from favorite artists.
- **Cora**, **Citeseer** and **PubMed** [35] are citation networks, where nodes stand for documents and edges denote citations. Each node has a bag-of-words feature vector and is labeled by the document category.

Following prior works [13], [15], we arbitrarily mask 10% of existing links within each graph as sensitive links and randomly select an equivalent quantity of nonexistent links as negative samples for evaluation.

2) Sensitive Link Inference Attacks and Privacy Metric: To assess the privacy protection efficacy of the PPGSL, we employ diverse methods to conduct various sensitive link inference attacks on its generated published graph. A lower inference success rate indicates stronger privacy protection. We consider eight distinct attacks that fall into the following two categories:

- **Structure-based attacks:** Common Neighbor (CN), Adamic-Adar (AA), Resource Allocation (RA), and SEAL [36]. These methods calculate node proximity scores as probabilities for the existence of sensitive links. The three traditional metrics are presented in Table III. SEAL extracts local subgraphs around each target link and trains a link prediction model based on them. We implement SEAL with node attributes disregarded and the hop number set to 2.
- **Embedding-based attacks:** node2vec [22] with cosine similarity (N2V+sim), GAE [37] with cosine sim-

[5]All datasets are available from the PyTorch Geometric libraries: https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html.

ilarity (GAE+sim), and combinations with LinearSVC (N2V+ML, GAE+ML). These methods first obtain node embeddings through graph representation learning and then compute the probability of sensitive link existence via similarity metrics (*e.g.*, cosine similarity [13], [14]) or classifiers (*e.g.*, LinearSVC [38], [39]). For classifiers, the input features are derived from concatenating the embeddings of two nodes, using published graph links as positive samples and an equal number of unconnected node pairs as negative samples for training.

In alignment with previous works [13], [40], we adopt the AUC (area under the ROC curve) as the privacy metric. A higher AUC indicates that it is easier for an attacker to infer sensitive links, leading to higher privacy risks and worse privacy protection effect.

3) Utility Evaluation Tasks and Metrics: We leverage two widely used graph-based tasks, *i.e.,* link prediction and node classification, to evaluate the utility of the published graph.

Link prediction attempts to predict the missing or potential links on a graph [36], [41]. We randomly mask 10% of the real links from the original graph as positive testing links and sample the same number of nonexistent links as negative testing links. The set of testing links is nonoverlapping with the set of sensitive links. For a published graph, we train a GNN model via unsupervised loss to procure node embeddings [16] and then compute embedding similarity to predict the existence of testing links. We employ the AUC as the performance metric of link prediction.

Node classification aims to correctly classify the unlabeled nodes on the basis of the proportion of labeled nodes [14], [40]. We randomly split the nodes of the original graph, allotting 30% of the nodes for training and reserving the remaining 70% for testing. Given a published graph, we train a two-layer GCN model [24] in a semisupervised manner [24] to predict node labels. The F1 score is used to measure the performance of node classification.

4) Baselines: We compare the privacy–utility trade-off performance of PPGSL with that of seven baselines:

Random [15] randomly removes partial links and adds the same number of new links to generate perturbed graphs.

DICE [42] generates perturbed graphs by deleting links connected to nodes with sensitive links and adding links between nodes without sensitive links.

PrivGraph [43] exploits community information to generate synthetic graphs with differential privacy guarantees.

EdgeRand [40] is an edge–DP defense strategy, which randomly flips each entry in the adjacency matrix according to a Bernoulli random variable.

LapGraph [40] also guarantees edge–DP. It precalculates the original graph density using a small privacy budget and uses that density to clip the perturbed adjacency matrix.

RW-LP [44] replaces real links with fake links between the starting and terminal nodes of random walks.

PPNE [13] samples some candidate node pairs for link perturbation and then iteratively selects the optimal pairs with a high privacy–utility trade-off to yield the perturbed graph.

5) Running Environment: Our experimental platform is a PC with an Intel i5 12600KF CPU (10 cores @ 3.7 GHz), 32 GB of RAM, and an NVIDIA RTX 4070Ti SUPER GPU (16 GB). The operation system is Ubuntu 22.04 LTS. We utilize Python 3.11, PyTorch 1.12.1, and PyTorch Geometric 2.3.1.

6) PPGSL Implementation: If not specified, we employ the PPGSL-skip ($\mu = 50$) and the PPGSL-sparse ($k = 1$) speed-up strategies. We adjust $\alpha \in [0, 0.01]$ to obtain graphs with varying levels of privacy preservation[6]. For the GNN encoder f_ϕ, we utilize a 2-layer GCN with hidden dimensions of [128, 64] and set the training epoch at 500. For the prediction head δ_φ, we choose cosine similarity[7]. For the graph learner $\mathcal{H}_\theta$, we fix the training epoch at 500 and choose the Adam optimizer with a learning rate of 0.5 to optimize the parameterized graph structure. We run all the experiments five times and report the average results.

B. Experimental Results

1) Privacy–utility Trade-off Performance Compared with Baselines: We report the privacy–utility trade-off performance of the PPGSL and five baselines under the utility tasks of link prediction (Fig. 2) and node classification (Fig. 3). In this part, we employ GAE+sim as the attack method because of its widespread use and superior attack performance [13], [14]. We plot two variants of the PPGSL where $k = 0$ or $k = 1$: the PPGSL ($k = 0$) signifies the parameterization solely of existing links, confining the structure perturbation to edge deletion; the PPGSL ($k = 1$) also parameterizes a subset of nonexistent links, expanding the structure perturbation to encompass both edge deletion and addition.

Herein, a low value on the y-axis means higher privacy protection (lower attack performance), whereas a high value on the x-axis indicates better utility. We see that directly releasing original graphs poses a significant risk of privacy leakage for sensitive links. The points at the top right of the curves denote the attack AUC and utility evaluations on the original graphs, where these attacks achieve extremely high attack AUC scores.

Note that by modifying the hyperparameters to control the privacy–utility trade-off, we can draw a line for each method to show how utility changes with varying levels of privacy protection. Most baselines exhibit limited privacy protection capability, with the lowest attack AUC failing to reach 50% (almost equivalent to random guessing, representing optimal privacy protection). In contrast, the PPGSL ($k = 1$) easily achieves an attack AUC of approximately 50%, demonstrating strong privacy-safeguarding ability.

With respect to the performance of the privacy–utility trade-off, the PPGSL and the baselines show the same pattern: with greater perturbation, the privacy attack becomes more difficult, and the utility decreases. More importantly, we observe that the PPGSL ($k = 0$) and the PPGSL ($k = 1$) usually provide greater privacy protection given the same utility, thereby achieving a better privacy–utility trade-off (*i.e.*, the PPGSL often appears in the lower right corner of the figures). Specifically, for *Cora* in Fig. 2, when the utility level (AUC

[6]Guidance for selecting α is provided in Appendix B.

[7]To demonstrate the robustness of our approach, we also evaluate its performance with various surrogate attack models. The results and analysis are presented in Appendix D.

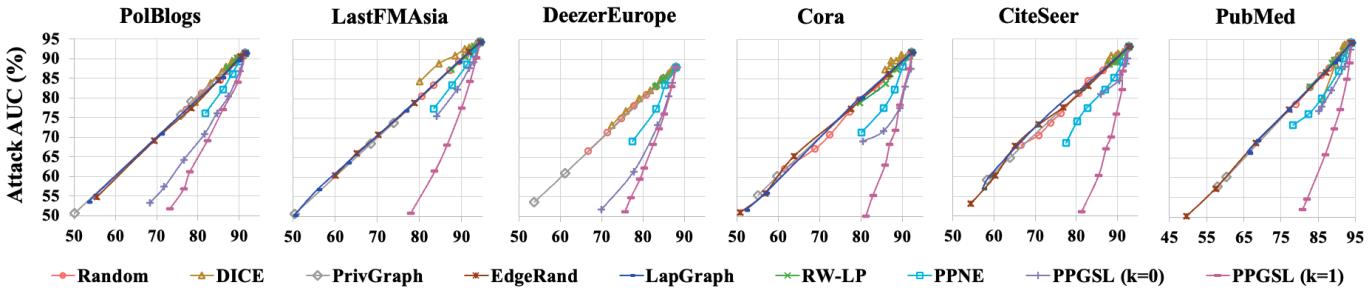


Fig. 2. Privacy–utility trade-off performance of the PPGSL. X-axis: AUC (%) of the utility task in terms of link prediction; Y-axis: AUC (%) of the GAE+sim attack method. Points at the top right of the curves represent evaluations of the original graph. The EdgeRand and LapGraph methods result in an out-of-memory error on *DeezerEurope*.

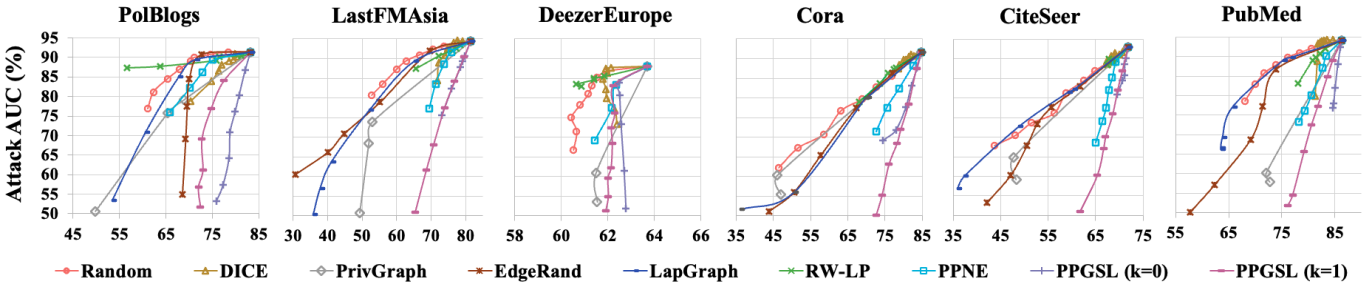


Fig. 3. Privacy–utility trade-off performance of the PPGSL. X-axis: F1 score (%) of the utility task in terms of node classification; Y-axis: AUC (%) of the GAE+sim attack method. Points at the top right of the curves indicate evaluations of the original graph. The EdgeRand and LapGraph methods result in an out-of-memory error on *DeezerEurope*.

of link prediction) is approximately 81%, the attack AUC of the PPGSL ($k = 1$) can be reduced to approximately 50%.

Moreover, we note that the PPGSL ($k = 0$) and the PPGSL ($k = 1$) exhibit divergent privacy–utility trade-off performances under different utility tasks. For link prediction (Fig. 2), the PPGSL ($k = 1$) presents the best privacy–utility trade-off across all six datasets; for node classification (Fig. 3), however, the PPGSL ($k = 0$) attains the best privacy–utility trade-off on more datasets than the PPGSL ($k = 1$) does. This disparity could be attributed to the fact that edge addition introduces greater disruption in node classification; thus, the PPGSL ($k = 1$) incurs greater utility loss when safeguarding privacy, whereas link prediction tasks are less susceptible to edge addition.

2) Privacy Protection Effects Against Various Inference Attacks: We conduct experiments to evaluate the PPGSL against eight sensitive link inference attacks, aiming to verify its generalized privacy protection effectiveness. Figs. 4 and 5 show the results on six datasets under the utility tasks of link prediction and node classification, respectively. Our results show that the PPGSL effectively defends against various inference attacks, as the PPGSL greatly decreases the attack AUC with tolerable utility loss on downstream tasks. Specifically, the PPGSL reduces the AUC of most attack methods to approximately 50% on *Cora* while preserving strong performances in link prediction (AUC of 81%) and node classification (F1 score of 73%). In addition, the fluctuation in node classification results on *DeezerEurope* is relatively minor. This can be attributed to the limited influence of graph structure information on this task for *DeezerEurope*, as an MLP model using solely node features achieves a sufficiently high F1 score of 63%.

3) Parameter Sensitivity: Sec. III-D introduces the PPGSL-sparse with a sampling factor k and the PPGSL-skip with an update interval μ to scale up the PPGSL. Here, we train the PPGSL with varying k and μ, and the results on *Cora* are shown in Fig. 6. The PPGSL-FGP uses the PPGSL's model architecture that parameterizes the full adjacency matrix. We observe that $k = 50$ and $\mu = 50/10/1$ yield similar trade-off performances, indicating that μ has little effect on the performance of the PPGSL. Both the PPGSL ($k = 1$) and the PPGSL ($k = 5$) demonstrate similar performance, significantly outperforming the PPGSL ($k = 0$) and the PPGSL-FGP. This is likely because the PPGSL ($k = 0$) removes only edges without adding any, resulting in minimal perturbation and a lower maximum level of privacy protection. In contrast, the PPGSL-FGP imposes no restrictions on the addition of edges, leading to greater perturbation and consequently greater utility loss. The PPGSL ($k = 1$) and the PPGSL ($k = 5$), however, limit the maximum number of added edges, resulting in moderate perturbation. They achieve a significant level of privacy protection while limiting utility loss, thus offering better privacy–utility trade-off performance. Hence, we recommend setting $k = 1$ and $\mu = 50$ in most cases (for node classification tasks, $k = 0$ can also be tried as discussed in Sec. V-B1).

4) Scalability of the PPGSL: We present the memory usage and training time of the PPGSL with simulated networks of diverse scales in Fig. 7. Specifically, we generate a series of Erdos–Renyi graphs with node counts ranging from 100 to 500,000 and an average node degree of 10. In general, the memory usage and training time of the PPGSL ($k = 0, \mu =$

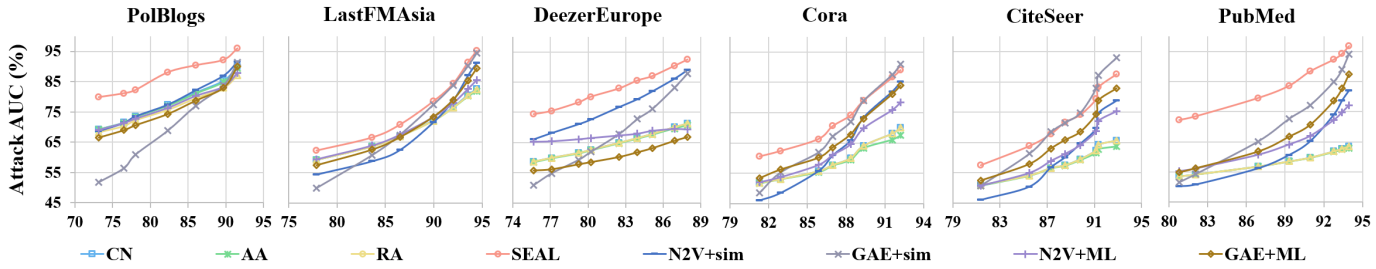


Fig. 4. Privacy protection effect of PPGSL against various attack methods under the link prediction utility task. X-axis: AUC (%) of link prediction.

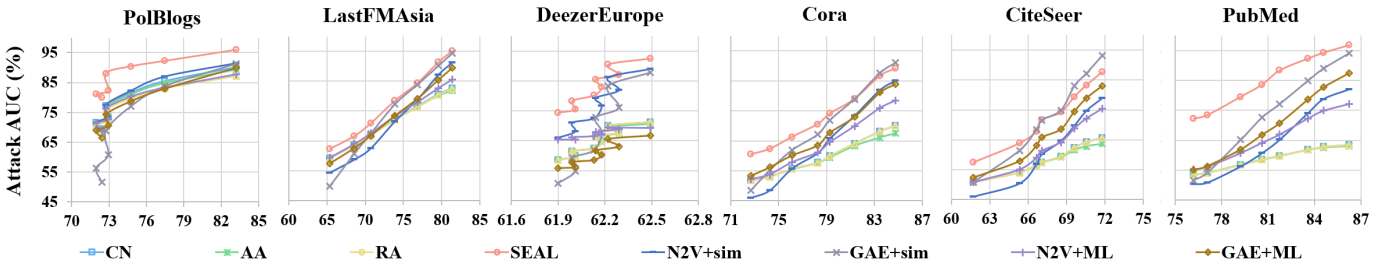


Fig. 5. Privacy protection effect of PPGSL against various attack methods under the node classification utility task. X-axis: F1 score (%) of node classification.

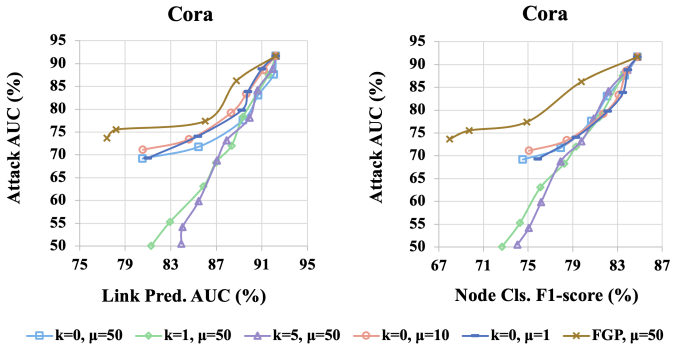


Fig. 6. Parameter analysis of k and μ on *Cora*. Left: link prediction utility task; right: node classification utility task.

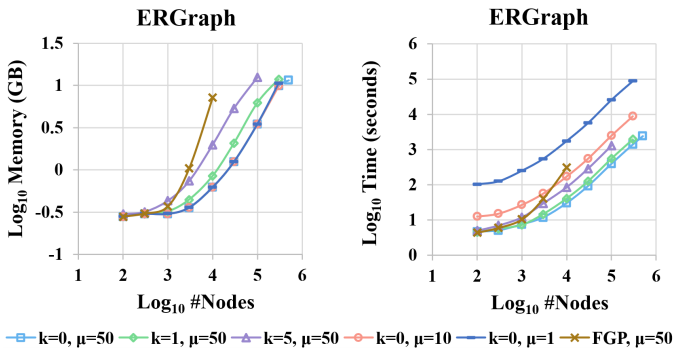


Fig. 7. Memory usage (left) and time consumption (right) for training the PPGSL on Erdos–Renyi graphs with varying node quantities. The training epoch is set to 500, as the PPGSL typically converges within 500 iterations.

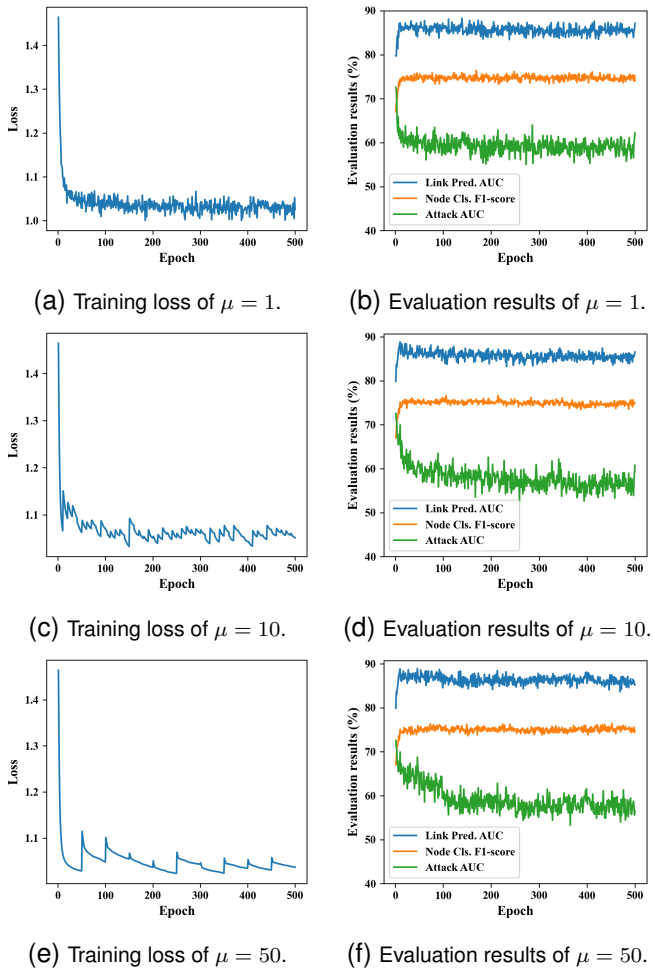
50) increase approximately linearly with the node count. With $\mu = 50$, we compare the PPGSL-sparse with $k \in \{0, 1, 5\}$ and the PPGSL-FGP. The PPGSL-sparse drastically reduces memory usage, particularly when k is small. With $k = 0$, we compare the PPGSL-skip with $\mu \in \{1, 10, 50\}$, and it is apparent that as μ increases, the time consumption diminishes remarkably, whereas the privacy–utility trade-off performance remains almost unchanged (please see Fig. 6).

Notably, for a graph with up to 100,000 nodes, the PPGSL can complete training with our recommended settings ($k = 1$, $\mu = 50$) in just 9.2 minutes, with an average memory usage of 6.2 GB. In contrast, for a graph of the same size, the most comparable method, PPNE, requires approximately 16.7 minutes per iteration and converges around the 3,000th iteration [13]. Consequently, the PPGSL achieves a training speedup of $16.7 \times 3,000 \div 9.2 \approx 5,445.7$ times faster than the PPNE for a complete training process.

5) Convergence of the PPGSL: We conduct experiments to verify the convergence of PPGSL. Fig. 8 illustrates the changes in loss and privacy/utility evaluation results during the training of the graph learner $\mathcal{H}_\theta$ for different values of μ (the update interval of the surrogate attack model). We observe that training loss generally decreases, with minor fluctuations when $\mu \neq 1$. In particular, when $\mu = 50$, the loss jumps every 50 epochs (see Fig. 8e) due to updates of the surrogate attack model. At these points, the surrogate model changes to a stronger version, temporarily increasing the privacy loss.

Regarding the privacy/utility evaluation results (Fig. 8b, 8d, and 8f), we observe that as the number of epochs increases, both privacy protection (as indicated by a general decrease in the attack AUC) and utility maintenance (reflected in the increase in the link prediction AUC and node classification F1 score) improve. Therefore, the PPGSL demonstrates fast and stable convergence capability, ensuring the effectiveness of the privacy-preserving graph learning process. Additionally, the convergence speed is higher when μ is smaller.

6) Comparison of Different Training Protocols: We compare our proposed SITP (with an update interval of $\mu = 50$) against a standard adversarial training baseline, ADV, which follows the protocol of AdvReg [30]. As shown in Fig. 9, SITP demonstrates a superior privacy–utility trade-off. Compared with the ADV baseline, it simultaneously achieves a higher

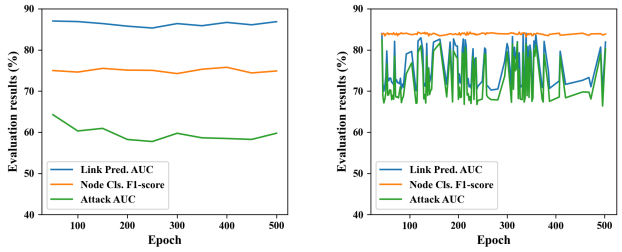


(a) Training loss of $\mu = 1$. (b) Evaluation results of $\mu = 1$.

(c) Training loss of $\mu = 10$. (d) Evaluation results of $\mu = 10$.

(e) Training loss of $\mu = 50$. (f) Evaluation results of $\mu = 50$.

Fig. 8. Variations in loss and privacy/utility evaluation results as the number of epochs increases in the training process of graph learner $\mathcal{H}_\theta$ on *Cora* under different μ settings.



(a) Evaluation results under SITP. (b) Evaluation results under ADV.

Fig. 9. Comparison of privacy/utility evaluation results across different training protocols. SITP: our proposed secure iterative training protocol, with a surrogate attack model update interval of $\mu = 50$; ADV: typical adversarial training protocols, such as AdvReg [30].

link prediction AUC (better utility) and a lower attack AUC (stronger privacy). A detailed comparison of convergence and training efficiency is provided in Appendix C.

VI. Related Works

A. Privacy-Preserving Graph Data Publishing

We first review three typical categories of privacy-preserving graph data publishing studies aimed at protecting link privacy: graph anonymization methods, differential privacy (DP) methods, and heuristic-based methods. Graph anonymization methods [44]–[50] aim to anonymize the links on the original graph by rewiring them via randomization techniques while preserving certain properties of the original graph to ensure utility. However, the primary goal of anonymization is to protect links on the original graph from reidentification rather than safeguarding hidden sensitive links against inference attacks.

DP methods [43], [51]–[54] are conventionally designed to defend against Bayesian inference attacks without arbitrary prior knowledge. In the context of graph privacy protection, DP methods usually prevent the original graph from being inferred on the basis of the released graph or embedding by introducing random noise with rigorous theoretical privacy guarantees. However, they often need to introduce excessive noise into the original data to defend against various link inference attacks (*e.g.*, GNN attacks [14], [55]), making it challenging to achieve a good privacy–utility trade-off.

Several heuristic-based methods [13], [15] have been proposed to defend against link inference attacks. They often first assess the privacy and utility of potential perturbations in a graph; then, they manually select the perturbation that provides the best privacy–utility trade-off to find the near-optimal graph structure iteratively. However, these methods often result in significant computational costs and produce only local optimal solutions rather than global optimal ones. They barely enumerate all the potential perturbations for privacy and utility assessment.

Compared with existing methods, the PPGSL is designed with a crafted privacy objective to measure the privacy leakage caused by sensitive link inference attacks. Additionally, the PPGSL automatically optimizes the graph structure to achieve an optimal privacy–utility trade-off via a learning method.

B. Link Inference Attacks

Link inference attacks exploit inherent patterns within the graph to reidentify or infer private structural information. Depending on the goal, method, and attacker's knowledge, these attacks can be divided into two categories: attacks in graph data publishing and attacks against open GNN APIs.

Link inference attacks in graph data publishing. In this context, attackers have access to the whole published graph data [15] or its node embedding [10]. Specifically, if attackers obtain the node embedding of the target graph, they can directly conduct embedding-based attacks by leveraging the similarity information between the embeddings of target node pairs [10], [13], [56]. Comparatively, given the published graph data, attackers can carry out link inference attacks via Bayesian methods and structure-based approaches [9], [36], [57] or conduct embedding-based attacks after learning node embedding with the published graph [13].

Link inference attacks against open GNN APIs. This stream of studies typically assumes that there is an open GNN API, which is trained to complete the node classification task, and attackers have black-box access to this API [14], [40], [58], [59]. Recently, the LinkTeller attack method was

introduced, which recovers private links on a graph through influence analysis [40]. Specifically, by querying the GNN API with adversarial input node features and analyzing its influence on the target node's output, an attacker can infer whether a link exists between the input node and the target node. Moreover, some recent work has considered the disparity in individual privacy risks [59], [60].

This work focuses on protecting sensitive links from inference attacks in the context of whole graph data sharing. We also note that some existing works have proposed defense strategies regarding attacks against open GNN APIs [58], [61]. Since these strategies typically aim to build privacy-preserving GNN models, they are unsuitable for the graph data sharing problem we address.

C. Graph Structure Learning

Graph structure learning (GSL) seeks to simultaneously derive an optimized graph structure and corresponding graph representations [17]. Most current GSL studies focus on learning robust graph representations or improving the performance on specific downstream tasks through the concurrent optimization of the graph structure and the GNN encoder [18], [19], [25], [62]–[65]. Recently, self-supervised GSL has emerged to address scenarios where the task label is scarce or downstream tasks are unknown [26], [66], [67].

Although both the PPGSL and GSL procure the targeted graph via a learning-based approach, the PPGSL significantly deviates from conventional GSL methods in its training objectives and training protocol. Specifically, while most GSL methods aim to learn a robust GNN encoder, our goal is to learn privacy-preserving graph data for publishing. The components of the PPGSL and GSL frameworks differ. Notably, the PPGSL incorporates surrogate attack models to generate privacy protection signals—a component that is absent in standard GSL approaches. The introduction of this new component also necessitates an effective and distinct training protocol.

VII. Conclusion and Discussion

This work makes theoretical contributions to the field of trustworthy graph systems by expanding the research scope and supplying novel privacy-preserving technologies [68]. Specifically, we formulate a novel learning-based graph data publishing problem against sensitive link inference attacks and propose a privacy-preserving graph structure learning framework, dubbed PPGSL. Two core modules are designed to parameterize the graph structure and optimize for privacy and utility objectives; a secure iterative training protocol is introduced to ensure privacy preservation and stable convergence. Theoretical analyses validate the convergence and optimality of the PPGSL. Extensive experiments demonstrate the state-of-the-art performance of the PPGSL in achieving an optimal privacy–utility trade-off.

We also discuss the limitations and future directions of this work. First, the PPGSL currently focuses only on perturbing the graph structure to protect sensitive links. Looking forward, we aim to design a unified framework that can learn both node features and topological structure for privacy-preserving data publishing. Second, we adopt a GNN-based surrogate attack model in our method, which may not represent all adversaries with different knowledge. Devising a more powerful surrogate attack model to capture broader privacy risks is a promising future research direction. Finally, while the PPGSL seeks to reduce the average privacy risk across the entire graph, it is crucial to consider the disparity in individual risks to ensure fairness [59], [60]. Incorporating both privacy and fairness considerations can lead the PPGSL toward more comprehensive and trustworthy data sharing [69].

References

[1] X. Han, L. Wang, N. Crespi, S. Park, and Á. Cuevas, "Alike people, alike interests? inferring interest similarity in online social networks," *Decis. Support Syst.*, vol. 69, pp. 92–106, 2015.

[2] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The World Wide Web Conference*, 2019, pp. 417–426.

[3] J. Suárez-Varela, P. Almasan, M. Ferriol-Galmés, K. Rusek, F. Geyer, X. Cheng, X. Shi, S. Xiao, F. Scarselli, A. Cabellos-Aparicio *et al.*, "Graph neural networks for communication networks: Context, use cases and opportunities," *IEEE Network*, vol. 37, no. 3, pp. 146–153, 2021.

[4] L. Wang, D. Chai, X. Liu, L. Chen, and K. Chen, "Exploring the generalizability of spatio-temporal traffic prediction: Meta-modeling and an analytic framework," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3870–3884, 2021.

[5] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.

[6] "AMiner Dataset." [Online]. Available: https://www.aminer.cn/aminer_data

[7] "Facebook Graph API." [Online]. Available: https://developers.facebook.com/docs/graph-api

[8] J. Jia, B. Wang, L. Zhang, and N. Z. Gong, "Attriinfer: Inferring user attributes in online social networks using markov random fields," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 1561–1569.

[9] Y. Zhang, M. Humbert, B. Surma, P. Manoharan, J. Vreeken, and M. Backes, "Towards plausible graph anonymization," in *27th Annual Network and Distributed System Security Symposium*, 2020.

[10] V. Duddu, A. Boutet, and V. Shejwalkar, "Quantifying privacy leakage in graph embedding," in *MobiQuitous 2020-17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2020, pp. 76–85.

[11] "General Data Protection Regulation (GDPR)." [Online]. Available: https://gdpr-info.eu/

[12] L. Yao, Z. Chen, X. Wang, D. Liu, and G. Wu, "Sensitive label privacy preservation with anatomization for data publishing," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 2, pp. 904–917, 2019.

[13] X. Han, Y. Yang, L. Wang, and J. Wu, "Privacy-preserving network embedding against private link inference attacks," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 2, pp. 847–859, 2024.

[14] X. He, J. Jia, M. Backes, N. Z. Gong, and Y. Zhang, "Stealing links from graph neural networks." in *USENIX Secur. Symp.*, 2021, pp. 2669–2686.

[15] S. Yu, M. Zhao, C. Fu, J. Zheng, H. Huang, X. Shu, Q. Xuan, and G. Chen, "Target defense against link-prediction-based attacks via evolutionary perturbations," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 2, pp. 754–767, 2019.

[16] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Adv. Neural Inf. Process. Syst.*, pp. 1024–1034, 2017.

[17] Y. Zhu, W. Xu, J. Zhang, Y. Du, J. Zhang, Q. Liu, C. Yang, and S. Wu, "A survey on graph structure learning: Progress and opportunities," *arXiv e-prints*, pp. arXiv–2103, 2021.

[18] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 66–74.

[19] N. Liu, X. Wang, L. Wu, Y. Chen, X. Guo, and C. Shi, "Compact graph structure learning via mutual information compression," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1601–1610.

[20] Z. Alkhalil, C. Hewage, L. Nawaf, and I. Khan, "Phishing attacks: A recent comprehensive study and a new anatomy," *Front. Comput. Sci.*, vol. 3, p. 563060, 2021.

[21] P. W. Fong, "Preventing sybil attacks by privilege attenuation: A design principle for social network systems," in *2011 IEEE Symposium on Security and Privacy*, 2011, pp. 263–278.

[22] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 855–864.

[23] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2019.

[24] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," arXiv preprint arXiv:1609.02907, 2016.

[25] B. Fatemi, L. El Asri, and S. M. Kazemi, "Slaps: Self-supervision improves structure learning for graph neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 22 667–22 681, 2021.

[26] Y. Liu, Y. Zheng, D. Zhang, H. Chen, H. Peng, and S. Pan, "Towards unsupervised deep graph structure learning," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1392–1403.

[27] D. Yang, D. Zhang, B. Qu, and P. Cudré-Mauroux, "Privcheck: Privacy-preserving check-in data publishing for personalized location based services," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2016, pp. 545–556.

[28] J. Jia and N. Z. Gong, "Attriguard: A practical defense against attribute inference attacks via adversarial machine learning," in *27th USENIX Security Symposium*, 2018, pp. 513–529.

[29] I.-C. Hsieh and C.-T. Li, "Netfense: Adversarial defenses against privacy attacks on neural networks for graph data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 796–809, 2023.

[30] M. Nasr, R. Shokri, and A. Houmansadr, "Machine learning with membership privacy using adversarial regularization," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 634–646.

[31] F. du Pin Calmon and N. Fawaz, "Privacy against statistical inference," in *50th Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 2012, pp. 1401–1408.

[32] X. Han, Y. Yang, J. Wu, and H. Xiong, "Hyobscure: Hybrid obscuring for privacy-preserving data publishing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 8, pp. 3893–3905, 2023.

[33] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 us election: Divided they blog," in *Proceedings of the 3rd International Workshop on Link Discovery*, 2005, pp. 36–43.

[34] B. Rozemberczki and R. Sarkar, "Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 1325–1334.

[35] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *International Conference on Machine Learning*. PMLR, 2016, pp. 40–48.

[36] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[37] T. N. Kipf and M. Welling, "Variational graph auto-encoders," arXiv preprint arXiv:1611.07308, 2016.

[38] F.-Y. Sun, J. Hoffman, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," in *International Conference on Learning Representations*, 2019.

[39] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 5812–5823, 2020.

[40] F. Wu, Y. Long, C. Zhang, and B. Li, "Linkteller: Recovering private edges from graph neural networks via influence analysis," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 2005–2024.

[41] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: Stat. Mech. Appl.*, vol. 390, no. 6, pp. 1150–1170, 2011.

[42] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *International Conference on Learning Representations*, 2019.

[43] Q. Yuan, Z. Zhang, L. Du, M. Chen, P. Cheng, and M. Sun, "Privgraph: Differentially private graph data publication by exploiting community information," in *USENIX Security*, 2023.

[44] P. Mittal, C. Papamanthou, and D. X. Song, "Preserving link privacy in social network based systems," in *20th Annual Network and Distributed System Security Symposium*, 2013.

[45] E. Zheleva and L. Getoor, "Preserving the privacy of sensitive relationships in graph data," in *Privacy, Security, and Trust in KDD: First ACM SIGKDD International Workshop*. Springer, 2008, pp. 153–171.

[46] X. Ying and X. Wu, "On link privacy in randomizing social networks," *Knowl. Inf. Syst.*, vol. 28, pp. 645–663, 2011.

[47] ——, "Randomizing social networks: a spectrum preserving approach," in *proceedings of the 2008 SIAM International Conference on Data Mining*. SIAM, 2008, pp. 739–750.

[48] A. M. Fard, K. Wang, and P. S. Yu, "Limiting link disclosure in social network analysis through subgraph-wise perturbation," in *Proceedings of the 15th International Conference on Extending Database Technology*, 2012, pp. 109–119.

[49] A. Milani Fard and K. Wang, "Neighborhood randomization for link privacy in social network analysis," *World Wide Web*, vol. 18, pp. 9–32, 2015.

[50] Y. Liu, S. Ji, and P. Mittal, "Smartwalk: Enhancing social network security via adaptive random walks," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 492–503.

[51] H. H. Nguyen, A. Imine, and M. Rusinowitch, "Differentially private publication of social graphs at linear cost," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, 2015, pp. 596–599.

[52] R. Chen, B. C. Fung, P. S. Yu, and B. C. Desai, "Correlated network data publication via differential privacy," *VLDB J.*, vol. 23, pp. 653–676, 2014.

[53] Q. Xiao, R. Chen, and K.-L. Tan, "Differentially private network data release via structural inference," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 911–920.

[54] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, "Generating synthetic decentralized social graphs with local differential privacy," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 425–438.

[55] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *USENIX Security Symposium*, pp. 1895–1912, 2019.

[56] X. Wang and W. H. Wang, "Link membership inference attacks against unsupervised graph representation learning," in *Proceedings of the 39th Annual Computer Security Applications Conference*, 2023, pp. 477–491.

[57] X. Xian, T. Wu, Y. Liu, W. Wang, C. Wang, G. Xu, and Y. Xiao, "Towards link inference attack against network structure perturbation," *Knowl.-Based Syst.*, vol. 218, p. 106674, 2021.

[58] L. Meng, Y. Bai, Y. Chen, Y. Hu, W. Xu, and H. Weng, "Devil in disguise: Breaching graph neural networks privacy through infiltration," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 1153–1167.

[59] H. Zhang, B. Wu, S. Wang, X. Yang, M. Xue, S. Pan, and X. Yuan, "Demystifying uneven vulnerability of link stealing attacks against graph neural networks," in *International Conference on Machine Learning*. PMLR, 2023, pp. 41 737–41 752.

[60] H. Zhang, X. Yuan, and S. Pan, "Unraveling privacy risks of individual fairness in graph neural networks," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2024, pp. 1712–1725.

[61] S. Sajadmanesh, A. S. Shamsabadi, A. Bellet, and D. Gatica-Perez, "{GAP}: Differentially private graph neural networks with aggregation perturbation," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 3223–3240.

[62] Y. Chen, L. Wu, and M. Zaki, "Iterative deep graph learning for graph neural networks: Better and robust node embeddings," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 19 314–19 326, 2020.

[63] D. Luo, W. Cheng, W. Yu, B. Zong, J. Ni, H. Chen, and X. Zhang, "Learning to drop: Robust graph neural network via topological denoising," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021, pp. 779–787.

[64] Q. Sun, J. Li, H. Peng, J. Wu, X. Fu, C. Ji, and S. Y. Philip, "Graph structure learning with variational information bottleneck," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, 2022, pp. 4165–4174.

[65] H. Wang, Y. Fu, T. Yu, L. Hu, W. Jiang, and S. Pu, "Prose: Graph structure learning via progressive strategy," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 2337–2348.

[66] K. Li, Y. Liu, X. Ao, J. Chi, J. Feng, H. Yang, and Q. He, "Reliable representations make a stronger defender: Unsupervised structure refinement for robust gnn," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 925–935.

[67] J. Zhao, Q. Wen, M. Ju, C. Zhang, and Y. Ye, "Self-supervised graph structure refinement for graph neural networks," in *Proceedings of*

the Sixteenth ACM International Conference on Web Search and Data Mining, 2023, pp. 159–167.
[68] B. Wu, Y. Bian, H. Zhang, J. Li, J. Yu, L. Chen, C. Chen, and J. Huang, "Trustworthy graph learning: Reliability, explainability, and privacy protection," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 4838–4839.
[69] H. Zhang, B. Wu, X. Yuan, S. Pan, H. Tong, and J. Pei, "Trustworthy graph neural networks: Aspects, methods, and trends," *Proceedings of the IEEE*, vol. 112, no. 2, pp. 97–139, 2024.
[70] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," in *International Conference on Learning Representations*, 2017.
[71] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm, "Mutual information neural estimation," in *International Conference on Machine Learning*. PMLR, 2018, pp. 531–540.
[72] P. Cheng, W. Hao, S. Dai, J. Liu, Z. Gan, and L. Carin, "Club: A contrastive log-ratio upper bound of mutual information," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1779–1788.
[73] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," in *International Conference on Learning Representations*, 2019.
[74] B. Poole, S. Ozair, A. Van Den Oord, A. Alemi, and G. Tucker, "On variational bounds of mutual information," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5171–5180.
[75] B. Wang, J. Guo, A. Li, Y. Chen, and H. Li, "Privacy-preserving representation learning on graphs: A mutual information perspective," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 1667–1676.

Yucheng Wu received the bachelor's degree in data science and big data technology from Shanghai University of Finance and Economics, China, in 2023. She is currently working toward the PhD degree in computer software and theory with the School of Computer Science, Peking University, China. Her research interests include large-scale data mining and graph neural networks.

Yuncong Yang received a BS, MS and Ph.D. degree in management science and engineering from the Shanghai University of Finance and Economics, in 2018, 2020 and 2025, respectively. He is a member of the Key Laboratory of Interdisciplinary Research of Computation and Economics (Shanghai University Finance and Economics), Ministry of Education. His work has been published in leading journals in computer science, including TDSC and TKDE. His research interests include graph neural networks and privacy protection in machine learning.

Xiao Han is a full professor at the School of Economics and Management, Beihang University. She received a Ph.D. in informatics from the Pierre and Marie Curie University and Institut Mines-TELECOM/TELECOM SudParis in 2015. Her research focuses on data-driven intelligent systems in business and societal contexts, with particular focuses on data security and privacy. Her work has been published in leading journals and top-tier conference proceedings in information systems and computer science, including MISQ, JOC, TDSC, TIFS, TKDE, TSE, WWW, AAAI, etc.

Leye Wang is a tenured associate professor at Key Lab of High Confidence Software Technologies (Peking University), Ministry of Education, China, and School of Computer Science, Peking University. His research interests include ubiquitous computing and data privacy protection. Wang received a Ph.D. in computer science from the Pierre and Marie Curie University and Institut Mines-TELECOM/TELECOM SudParis, France, in 2016. His research has appeared in journals and conference proceedings such as MISQ, JOC, TDSC, TIFS, TKDE, AI Journal, IEEE Computer, IEEE Comm. Mag., WWW, AAAI, ASE, etc.computing, mobile crowdsensing, and urban computing.

Junjie Wu is currently the full professor of the School of Economics and Management, Beihang University. He is also the director of the MIIT Key Laboratory of Data Intelligence and Management. He holds a B.E. degree from the School of Civil Engineering and a Ph.D. degree from the School of Economics and Management, Tsinghua University. His research interests include data and decision intelligence, with intense applications to business, finance, cities and industries. He has published prolifically in journals and proceedings including MISQ, ISR, JOC, TKDE, KDD, NeurIPS, AAAI, etc.

APPENDIX

A. Proof of Theoretical Analysis

1) Convergence of the PPGSL:

Lemma 1. *Suppose that $\mathcal{L}(\theta_1, \theta_2)$ is strongly convex with respect to θ_1 for any fixed θ_2, and that $\mathcal{L}$ is continuously differentiable with respect to both θ_1 and θ_2. Then the optimal solution θ_1^* changes slowly as θ_2 varies.*

Proof. Define $\theta_1^*(\theta_2) = \arg\min_{\theta_1} \mathcal{L}(\theta_1, \theta_2)$ as the optimal value of θ_1 given a fixed θ_2. Since $\mathcal{L}(\theta_1, \theta_2)$ is strongly convex in θ_1 for each fixed θ_2, $\theta_1^*(\theta_2)$ exists uniquely.

By the first-order optimality condition for $\theta_1^*(\theta_2)$, we have:

$$\nabla_{\theta_1} \mathcal{L}(\theta_1^*(\theta_2), \theta_2) = 0 \tag{14}$$

Since $\mathcal{L}$ is strongly convex in θ_1, the Hessian $\nabla^2_{\theta_1} \mathcal{L}(\theta_1^*(\theta_2), \theta_2)$ is positive definite for each θ_2. By the implicit function theorem, $\theta_1^*(\theta_2)$ is a continuously differentiable function of θ_2.

Taking the total derivative of $\nabla_{\theta_1} \mathcal{L}(\theta_1^*(\theta_2), \theta_2) = 0$ with respect to θ_2, we obtain the following:

$$\begin{aligned} \frac{d}{d\theta_2} \nabla_{\theta_1} \mathcal{L}(\theta_1^*(\theta_2), \theta_2) = \nabla^2_{\theta_1} \mathcal{L}(\theta_1^*(\theta_2), \theta_2) \cdot \frac{d\theta_1^*(\theta_2)}{d\theta_2} \\ + \nabla_{\theta_1} \nabla_{\theta_2} \mathcal{L}(\theta_1^*(\theta_2), \theta_2) = 0 \end{aligned} \tag{15}$$

Solving for $\frac{d\theta_1^*(\theta_2)}{d\theta_2}$, we obtain the following:

$$\frac{d\theta_1^*(\theta_2)}{d\theta_2} = -\left(\nabla^2_{\theta_1} \mathcal{L}(\theta_1^*(\theta_2), \theta_2)\right)^{-1} \nabla_{\theta_1} \nabla_{\theta_2} \mathcal{L}(\theta_1^*(\theta_2), \theta_2) \tag{16}$$

Since $\mathcal{L}$ is strongly convex with respect to θ_1, $\nabla^2_{\theta_1} \mathcal{L}(\theta_1^*(\theta_2), \theta_2)$ is bounded below by a positive constant m, giving the following:

$$\left\| \frac{d\theta_1^*(\theta_2)}{d\theta_2} \right\| \leq \frac{1}{m} \left\| \nabla_{\theta_1} \nabla_{\theta_2} \mathcal{L}(\theta_1^*(\theta_2), \theta_2) \right\| \tag{17}$$

Thus, as θ_2 varies, the change in $\theta_1^*(\theta_2)$ is bounded, implying that $\theta_1^*(\theta_2)$ varies slowly with respect to θ_2 if $\nabla_{\theta_1} \nabla_{\theta_2} \mathcal{L}$ is well behaved.

This bound on $\frac{d\theta_1^*(\theta_2)}{d\theta_2}$ implies that $\theta_1^*(\theta_2)$ changes gradually as θ_2 changes, ensuring that the updates to θ_2 do not cause large, abrupt changes in optimal θ_1. Hence, the update scheme where θ_1 is reoptimized for each update of θ_2 remains stable. □

This lemma provides the theoretical basis for the claim that, under strong convexity of $\mathcal{L}$ with respect to θ_1, the optimal θ_1 will vary smoothly as θ_2 is updated, making this training approach converge more stably.

Proposition 1. *Under the training of the PPGSL with SITP, the following inequality holds from the t-th iteration to the $(t+1)$-th iteration:*

$$\mathbb{E}[\mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t+1)})] \leq \mathbb{E}[\mathcal{L}_{learner}(\phi^{(t)}, \theta^{(t)})] \tag{18}$$

Proof. (1) Update Step for ϕ. At each iteration t, we first reinitialize ϕ and optimize it with $\theta^{(t)}$ fixed to minimize $\mathcal{L}_{attack}(\phi, \theta^{(t)})$. Since the reinitialization is random, we consider the expected value of $\mathcal{L}_{attack}$ after this step. Let $\mathbb{E}[\mathcal{L}_{attack}(\phi^{(t+1)}, \theta^{(t)})]$ represent the expected minimum value of $\mathcal{L}_{attack}$ after reinitialization and optimization over ϕ. By definition of $\phi^{(t+1)}$, we have the following:

$$\mathbb{E}[\mathcal{L}_{attack}(\phi^{(t+1)}, \theta^{(t)})] \leq \mathbb{E}[\mathcal{L}_{attack}(\phi^{(t)}, \theta^{(t)})] \tag{19}$$

As θ only updates one step in each iteration, $\mathcal{L}_{learner}(\phi, \theta)$ changes more drastically when θ changes, and the change in ϕ is relatively small when θ changes according to Lemma 1. Thus, the change in ϕ results in little change in $\mathcal{L}_{learner}(\phi, \theta)$, and we see it as not changing:

$$\mathbb{E}[\mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t)})] \doteq \mathbb{E}[\mathcal{L}_{learner}(\phi^{(t)}, \theta^{(t)})] \tag{20}$$

(2) Update Step for θ. Suppose that $\mathcal{L}_{learner}$ is convex in θ with a Lipschitz continuous gradient. We fix $\phi^{(t+1)}$ and update θ via a gradient descent step to minimize $\mathcal{L}_{learner}(\phi^{(t+1)}, \theta)$. Assuming that $\mathcal{L}_{learner}$ is convex in θ and has a Lipschitz continuous gradient, we can ensure that for a sufficiently small learning rate η, the update satisfies the following:

$$\begin{aligned} \mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t+1)}) \leq \mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t)}) \\ - \frac{\eta}{2} \|\nabla_\theta \mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t)})\|^2 \end{aligned} \tag{21}$$

To derive this, we apply the Lipschitz continuity property to expand $\mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t+1)})$ around $\theta^{(t)}$ as follows:

$$\begin{aligned} \mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t+1)}) \leq \mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t)}) \\ + \nabla_\theta \mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t)})^\top (\theta^{(t+1)} - \theta^{(t)}) \\ + \frac{L}{2} \|\theta^{(t+1)} - \theta^{(t)}\|^2 \end{aligned} \tag{22}$$

where L represents the Lipschitz constant of the gradient of the function $\mathcal{L}_{learner}(\phi, \theta)$ with respect to θ. This means that the gradient of $\mathcal{L}_{learner}$ with respect to θ does not change too quickly, which we can mathematically state as $\|\nabla_\theta \mathcal{L}_{learner}(\phi, \theta') - \nabla_\theta \mathcal{L}_{learner}(\phi, \theta)\| \leq L\|\theta' - \theta\|$ for any θ and θ'.

Substitute the gradient descent update $\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta \mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t)})$ into the following inequality:

$$\begin{aligned} \mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t+1)}) \leq \mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t)}) \\ - \eta \|\nabla_\theta \mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t)})\|^2 \\ + \frac{L\eta^2}{2} \|\nabla_\theta \mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t)})\|^2 \end{aligned} \tag{23}$$

The terms can be rearranged to obtain the following:

$$\begin{aligned} \mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t+1)}) \leq \mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t)}) \\ - \left(\eta - \frac{L\eta^2}{2}\right) \|\nabla_\theta \mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t)})\|^2 \end{aligned} \tag{24}$$

Choosing $\eta \leq \frac{1}{L}$ ensures that $\eta - \frac{L\eta^2}{2} \geq \frac{\eta}{2}$, yielding

$$\begin{aligned} \mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t+1)}) \leq \mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t)}) \\ - \frac{\eta}{2} \|\nabla_\theta \mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t)})\|^2 \end{aligned} \tag{25}$$

Thus, as expected, updating θ decreases $\mathcal{L}_{learner}$ as follows:

$$\mathbb{E}[\mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t+1)})] \leq \mathbb{E}[\mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t)})] \tag{26}$$

(3) Combining Updates. Combining Eq. 20 and Eq. 26, the following inequality holds from the t-th iteration to the $(t+1)$-th iteration:

$$\mathbb{E}[\mathcal{L}_{learner}(\phi^{(t+1)}, \theta^{(t+1)})] \leq \mathbb{E}[\mathcal{L}_{learner}(\phi^{(t)}, \theta^{(t)})] \quad (27)$$

□

2) Optimal Privacy–utility Trade-off of the PPGSL:

Proposition 2. *Consider the following two optimization problems:*

- *Constrained Optimization Problem:*

$$\min_{\theta} \mathcal{L}_{priv}(\theta) \quad s.t. \quad \mathcal{L}_{util}(\theta) \leq \epsilon \quad (28)$$

- *Unconstrained Regularization Problem:*

$$\min_{\theta} \mathcal{L}_{learner}(\theta) = \mathcal{L}_{priv}(\theta) + \alpha\, \mathcal{L}_{util}(\theta) \quad (29)$$

Assume that both $\mathcal{L}_{priv}(\theta)$ and $\mathcal{L}_{util}(\theta)$ are convex functions of θ and that the feasible region satisfies Slater's condition. Then, there must exist an $\alpha^* \geq 0$ such that if $\alpha = \alpha^*$, the optimal solutions of both the constrained and unconstrained problems are equivalent.

Proof. We begin by constructing the Lagrangian for the constrained problem with the Lagrange multiplier $\lambda \geq 0$:

$$\mathcal{L}_{lagr}(\theta, \lambda) = \mathcal{L}_{priv}(\theta) + \lambda\, (\mathcal{L}_{util}(\theta) - \epsilon) \quad (30)$$

Our goal is to solve the following:

$$\max_{\lambda \geq 0} \min_{\theta} \mathcal{L}_{lagr}(\theta, \lambda) \quad (31)$$

Karush–Kuhn–Tucker (KKT) Conditions: Given that $\mathcal{L}_{priv}(\theta)$ and $\mathcal{L}_{util}(\theta)$ are convex functions and that the feasible region satisfies Slater's condition, the KKT conditions are both necessary and sufficient for optimality. Let θ^* be the solution to the constrained problem. The KKT conditions for (θ^*, λ^*) are as follows:

- Primal feasibility: $\mathcal{L}_{util}(\theta^*) \leq \epsilon$.
- Dual feasibility: $\lambda^* \geq 0$.
- Stationarity: $\nabla_\theta \mathcal{L}_{priv}(\theta^*) + \lambda^* \nabla_\theta \mathcal{L}_{util}(\theta^*) = 0$.
- Complementary slackness: $\lambda^* (\mathcal{L}_{util}(\theta^*) - \epsilon) = 0$.

By the complementary slackness condition, we consider two possible cases:

- Case 1: $\mathcal{L}_{util}(\theta^*) = \epsilon$. Here, the constraint is active, meaning that $\mathcal{L}_{util}(\theta^*)$ reaches the upper limit ϵ. The KKT conditions guarantee the existence of $\lambda^* > 0$. On the basis of the stationarity condition, $\lambda^* = -\frac{\nabla_\theta \mathcal{L}_{priv}(\theta^*)}{\nabla_\theta \mathcal{L}_{util}(\theta^*)}$. The solution θ^* for the constrained problem also minimizes the objective $\mathcal{L}_{learner}(\theta) = \mathcal{L}_{priv}(\theta) + \alpha \mathcal{L}_{util}(\theta)$ when $\alpha^* = -\frac{\nabla_\theta \mathcal{L}_{priv}(\theta^*)}{\nabla_\theta \mathcal{L}_{util}(\theta^*)}$, as $\nabla_\theta \mathcal{L}_{priv}(\theta^*) + \alpha^* \nabla_\theta \mathcal{L}_{util}(\theta^*) = 0$ for the extreme point θ^*.
- Case 2: $\lambda^* = 0$. Here, the utility constraint is not binding, *i.e.*, $\mathcal{L}_{util}(\theta^*) < \epsilon$. Since $\lambda^* = 0$, minimizing the unconstrained problem is equivalent to minimizing $\mathcal{L}_{priv}(\theta)$ alone. This solution also satisfies the original constraint $\mathcal{L}_{util}(\theta^*) \leq \epsilon$ without requiring any additional penalty. Furthermore, for optimizing $\mathcal{L}_{learner}(\theta)$, minimizing $\mathcal{L}_{priv}(\theta)$ alone is equivalent to having $\alpha^* = 0$.

Therefore, under these conditions, there exists an $\alpha = \alpha^*$ such that the optimal solution θ^* of the constrained problem also minimizes the unconstrained regularization problem. The two problems are equivalent when:

$$\alpha = \alpha^*, \quad \text{where } \alpha^* \text{ satisfies the KKT conditions.} \quad (32)$$

□

3) Generalized Privacy Protection Performance of the PPGSL: The GNN encoder f_ϕ, parameterized by ϕ, maps x_u in the observational space to an embedding vector z_u in a latent space, *i.e.*, $z_i = f_\phi(x_i)$, and $Z = [z_1; z_2; \ldots; z_N] \in \mathbb{R}^{N \times d}$ is the node embedding matrix. Z' is the node embedding matrix of G', where $Z' = f_\phi(G')$. Here, f_ϕ is the GNN encoder of the surrogate attack model. Taking node embedding as a bridge, we can reformulate the privacy goal $\min_\theta I(G'; E_s)$ as follows[8]:

$$\text{Embedding Goal: } \max_{\phi} I(G'; Z') \quad (33a)$$

$$\text{New Privacy Goal: } \min_{\theta} I(Z'; E_s) \quad (33b)$$

Since the random variables in Eq. 33 are possibly high-dimensional and their posterior distributions are unknown, the mutual information terms are difficult to calculate. Motivated by existing mutual information neural estimation methods [70]–[75], we solve this challenge by translating the intractable mutual information terms into tractable terms by designing variational bounds.

Lemma 2. *Training neural networks f_ϕ with the objective function $\mathcal{L}_{attack}$ of the PPGSL (Eq. 5) is equivalent to achieving the embedding goal (Eq. 33a).*

Proof. For the mutual information term in Eq. 33a, we derive the following variational lower bound:

$$\begin{aligned}
&I(G'; Z') \\
=&I(w'_{ij}; z'_i, z'_j) \\
=&H(w'_{ij}) - H(w'_{ij} | z'_i, z'_j) \\
=&H(w'_{ij}) + \mathbb{E}_{p(z'_i, z'_j, w'_{ij})} \log p(w'_{ij} | z'_i, z'_j) \\
=&H(w'_{ij}) + \mathbb{E}_{p(z'_i, z'_j, w'_{ij})} KL\left(p(\cdot | z'_i, z'_j) || q_\varphi(\cdot | z'_i, z'_j)\right) \\
&\qquad + \mathbb{E}_{p(z'_i, z'_j, w'_{ij})} \log q_\varphi(w'_{ij} | z'_i, z'_j) \\
\geq&H(w'_{ij}) + \mathbb{E}_{p(z'_i, z'_j, w'_{ij})} \log q_\varphi(w'_{ij} | z'_i, z'_j) \\
:=&I_{vLB}(w'_{ij}; z'_i, z'_j)
\end{aligned} \quad (34)$$

where $KL\,(p(\cdot)||q(\cdot))$ is the Kullback–Leibler divergence between two distributions $p(\cdot)$ and $q(\cdot)$ and is nonnegative. q_φ is an (arbitrary) auxiliary posterior distribution. $I_{vLB}(w'_{ij}; z'_i, z'_j)$ is the variational lower bound of the mutual information term, and $H(w'_{ij})$ is a constant. Note that the lower bound is tight when the auxiliary distribution q_φ becomes the true posterior distribution p.

[8]Note that we slightly misuse the notations G and E_s for the sake of simplicity. They are initially employed to represent the original graph and the sensitive links, and we also use them to denote random variables here.

Our goal is to maximize the variational lower bound by estimating the auxiliary posterior distribution q_φ via a parameterized neural network. We parameterize q_φ via a link predictor δ_φ defined on the node representations. Specifically, we have

$$\begin{aligned}
&\max_{\phi}\max_{\varphi} I_{vLB}(w'_{ij};z'_i,z'_j)\\
\Leftrightarrow&\max_{\phi}\max_{\varphi}\mathbb{E}_{p(z'_i,z'_j,w'_{ij})}\log q_\varphi(w'_{ij}|z'_i,z'_j)\\
=&\max_{\phi}\max_{\varphi}\mathbb{E}_{p(z'_i,z'_j,w'_{ij})}\log q_\varphi(w'_{ij}|f_\phi(x'_i),f_\phi(x'_j))\\
\approx&\max_{\phi}\max_{\varphi}\sum_{\langle i,j\rangle\in E'_p\cup E'_n}-CE\left(\delta_\varphi(f_\phi(x'_i),f_\phi(x'_j)),w'_{ij}\right)\\
\Leftrightarrow&\min_{\phi}\min_{\varphi}\sum_{\langle i,j\rangle\in E'_p\cup E'_n}CE\left(\delta_\varphi(f_\phi(x'_i),f_\phi(x'_j)),w'_{ij}\right)
\end{aligned}\tag{35}$$

where $CE(\cdot,\cdot)$ denotes the cross-entropy function. In the learned graph G', E'_p is a set of sampled existing edges, and E'_n is a set of sampled nonexistent edges. The final objective function in Eq. 35 is just $\mathcal{L}_{attack}$ in Eq. 5. Therefore, by taking $\mathcal{L}_{attack}$ as an objective function and training the parameterized neural networks, we can achieve the embedding goal (Eq. 33a). □

Lemma 3. *Training neural networks $\mathcal{H}_\theta$ with the objective function $\mathcal{L}_{priv}$ of the PPGSL (Eq. 7) is equivalent to achieving the new privacy goal (Eq. 33b).*

Proof. We define $w^s_{ij}\in\{0,1\}$ as the adjacency matrix of sensitive links, where $w^s_{ij}=1$ if $\langle v_i,v_j\rangle\in E_s$ and $w^s_{ij}=0$ otherwise. The new privacy goal (Eq. 33b) can be specified as $\min_\theta I(w^s_{ij};z'_i,z'_j)$, and we derive the vCLUB inspired by [72] as follows:

$$\begin{aligned}
&I(E_s;Z')\\
=&I(w^s_{ij};z'_i,z'_j)\\
\le&I_{vCLUB}(w^s_{ij};z'_i,z'_j)\\
=&\mathbb{E}_{p(z'_i,z'_j,w^s_{ij})}\log q_\varphi(w^s_{ij}|z'_i,z'_j)\\
&-\mathbb{E}_{p(z'_i,z'_j)p(w^s_{ij})}\log q_\varphi(w^s_{ij}|z'_i,z'_j)
\end{aligned}\tag{36}$$

where $q_\varphi(w^s_{ij}|z'_i,z'_j)$ is an auxiliary distribution of $p(w^s_{ij}|z'_i,z'_j)$ that needs to satisfy the following condition:

$$\begin{aligned}
&KL\left(p(z'_i,z'_j,w^s_{ij})||q_\varphi(z'_i,z'_j,w^s_{ij})\right)\le\\
&\qquad KL\left(p(z'_i,z'_j)p(w^s_{ij})||q_\varphi(z'_i,z'_j,w^s_{ij})\right)
\end{aligned}\tag{37}$$

That is, I_{vCLUB} is a mutual information upper bound if the variational joint distribution $q_\varphi(z'_i,z'_j,w^s_{ij})$ is closer to the joint distribution $p(z'_i,z'_j,w^s_{ij})$ than to $p(z'_i,z'_j)p(w^s_{ij})$.

To achieve the above inequation, we need to minimize the KL-divergence as follows:

$$\begin{aligned}
&\min_{\varphi}KL\left(p(z'_i,z'_j,w^s_{ij})||q_\varphi(z'_i,z'_j,w^s_{ij})\right)\\
=&\min_{\varphi}KL\left(p(w^s_{ij}|z'_i,z'_j)||q_\varphi(w^s_{ij}|z'_i,z'_j)\right)\\
=&\min_{\varphi}\mathbb{E}_{p(z'_i,z'_j,w^s_{ij})}\log p(w^s_{ij}|z'_i,z'_j)\\
&\qquad-\mathbb{E}_{p(z'_i,z'_j,w^s_{ij})}\log q_\varphi(w^s_{ij}|z'_i,z'_j)\\
\Leftrightarrow&\max_{\varphi}\mathbb{E}_{p(z'_i,z'_j,w^s_{ij})}\log q_\varphi(w^s_{ij}|z'_i,z'_j)
\end{aligned}\tag{38}$$

Finally, our target to achieve Eq. 33b becomes the following adversarial training objective:

$$\begin{aligned}
&\min_{\theta}\min_{\varphi} I_{vCLUB}(w^s_{ij};z'_i,z'_j)\\
\Leftrightarrow&\min_{\theta}\max_{\varphi}\mathbb{E}_{p(z'_i,z'_j,w^s_{ij})}\log q_\varphi(w^s_{ij}|z'_i,z'_j)\\
\approx&\min_{\theta}\max_{\varphi}\sum_{\langle i,j\rangle\in E_s}-CE\left(\delta_\varphi(z'_i,z'_j),w^s_{ij}\right)\\
\approx&\min_{\theta}\max_{\varphi}\sum_{\langle i,j\rangle\in E_s}-CE\left(\delta_\varphi(z'_i,z'_j),1\right)
\end{aligned}\tag{39}$$

where $CE(\cdot,\cdot)$ denotes the cross-entropy function. Eq. 39 draws away the sensitive node pair embeddings, which is equivalent to our training objective $\mathcal{L}_{priv}$. Therefore, by taking $\mathcal{L}_{priv}$ as an objective function and training the parameterized neural networks, we can achieve the new privacy goal Eq. 33b. □

Lemma 4. *In every update step of $\mathcal{H}_\theta$, if f_ϕ is sufficiently retrained to convergence, then optimizing θ to minimize $I(Z';E_s)$ results in a decrease in $I(G';E_s)$.*

Proof. In each update step of the graph learner, we fully retrain f_ϕ until convergence. According to Lemma 2, training f_ϕ with the objective function $\mathcal{L}_{attack}$ will achieve the intermediate goal (Eq. 33b), *i.e.*, $I(G';Z')$ reaches its maximum value. Owing to the expressive limitations of GNN [23], we assume that the maximum value of $I(G';Z')$ remains unchanged. We prove that if $I(G';Z')$ does not change and $I(Z';E_s)$ decreases, then $I(G';E_s)$ will also decrease.

Let $H(\cdot)$ denote the information entropy of a random variable. Given that the variability space of the random variables G', E_s, and Z' is fixed, we assume that $H(G')$, $H(E_s)$, and $H(Z')$ remain constant. From the relationship of mutual information, we have the following:

$$I(Z';E_s)=I(Z';E_s|G')+I(Z';E_s;G')\tag{40}$$

Assuming that $I(Z';E_s;G')$ either increases or remains unchanged, we can deduce that $I(Z';E_s|G')$ must decrease. Since $I(Z';E_s|G')$ is bounded by $0\le I(Z';E_s|G')\le H(Z'|G')$, it cannot decrease indefinitely. After sufficient iterations of optimization, $I(Z';E_s|G')$ approaches zero and cannot decrease further. At this point, the assumption that $I(Z';E_s|G')$ decreases becomes invalid, leading to the conclusion that $I(Z';E_s;G')$ must decrease.

During the overall optimization process, since $I(G';Z')$ remains unchanged and $I(Z';E_s)$ decreases, we conclude that $I(Z';E_s;G')$ decreases. Consequently, we find that $I(E_s;G')=I(G';E_s|Z')+I(Z';E_s;G')$ decreases, thus confirming that the derivation holds. □

Proposition 3. *In PPGSL training process, the mutual information $I(G';E_s)$ decreases.*

Proof. From Lemmas 2 and 3, training the objective functions in the PPGSL increases $I(G';Z')$ and decreases $I(Z';E_s)$. According to Lemma 4, increasing $I(G';Z')$ and decreasing $I(Z';E_s)$ together lead to a decrease in $I(G';E_s)$ in the PPGSL training process. □

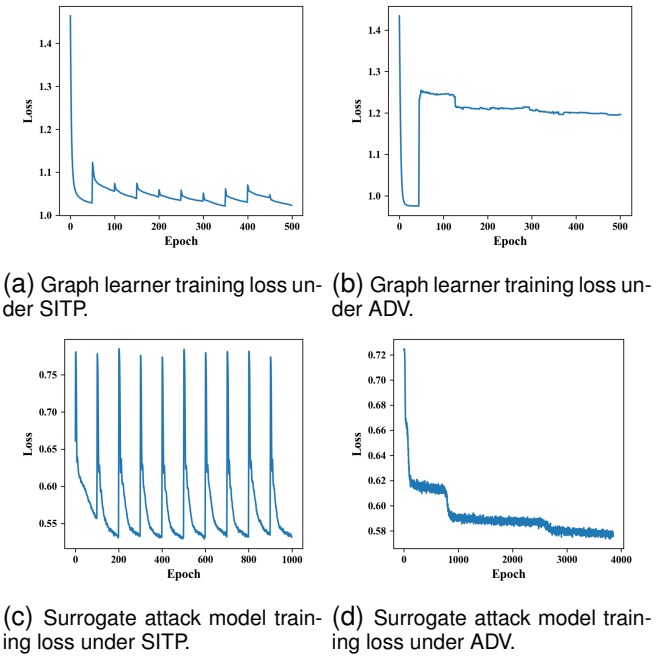


Fig. 10. Comparison of training robustness across different training protocols. SITP: our proposed secure iterative training protocol, with a surrogate attack model update interval of $\mu = 50$; ADV: typical adversarial training protocols, such as AdvReg [30].

B. Guidance on Trade-off Parameter Selection

Empirically, the trade-off parameter α is typically chosen within the range of [0.001, 0.01]. The appropriate value of α tends to be smaller for larger graphs (*i.e.,* graphs with more nodes) to achieve a similar level of privacy protection. This is because in larger graphs, the adjacency matrix has more entries. Consequently, for the same proportional perturbation, the Frobenius norm of the difference in adjacency matrices, and thus the utility loss $\mathcal{L}_{util}$, will be larger. A smaller α is therefore needed to balance $\mathcal{L}_{util}$ with the privacy loss term $\mathcal{L}_{priv}$. Moreover, to adjust the balance between utility and privacy, if a higher level of utility is desired, α can be appropriately increased; if a stronger privacy protection effect is needed, α can be appropriately decreased.

C. Comparison of Different Training Protocols

We compare the training robustness between our proposed SITP with an update interval $\mu = 50$ and typical adversarial training protocols similar to AdvReg [30] (denoted ADV). From Fig. 10, we can conclude that SITP outperforms ADV in terms of convergence and training efficiency.

- Convergence: Under SITP, the graph learner's training loss generally decreases over time, with minor jumps at multiples of 50 epochs due to updates in the surrogate attack model. In contrast, under ADV, the graph learner's training loss does not consistently decrease, which may converge to a local rather than a global optimum (compare Fig. 10a and 10b).
- Training efficiency: With SITP, when the graph learner updates for 500 epochs, the surrogate attack model only requires 1,000 total updates (due to its faster convergence, where retraining does not significantly increase the computation time). In contrast, ADV needs approximately 4,000 updates of the surrogate attack model for the same number of graph learner updates, resulting in higher training overhead (compare Fig. 10c and 10d).

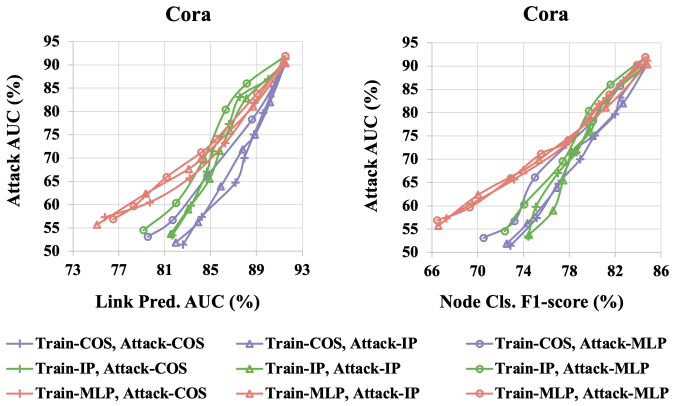


Fig. 11. Robustness across diverse surrogate models on *Cora*. COS: cosine similarity-based predictor, IP: inner production-based predictor, MLP: multilayer perceptron-based predictor.

D. Robustness across Diverse Surrogate Models

Most link prediction attacks operate on a common principle: they all exploit feature similarity and structural similarity (embedding similarity also results from feature and structural similarity). This shared foundation suggests that a defense trained against one type of surrogate model should be robust against others. To verify this hypothesis, we evaluate our defense against surrogate models built with three distinct prediction heads: a cosine similarity-based predictor (COS), an inner product-based predictor (IP), and a multilayer perceptron-based predictor (MLP). In our experiment, the defender trains the defense graph via a surrogate model with one specific predictor. The attacker, however, is free to use a model with any of the three predictors, creating both matched (defender and attacker use the same model) and mismatched scenarios. As shown in Fig. 11, our defense remains effective even in mismatched scenarios, with only slight performance degradation compared with when the models are matched.