Online Supplementary Materials

Appendix A. Literature Comparison

Table A1 Comparison of related studies based on their shared information

Algorithm perspective	Research	Information shared	Data-sharing-free method	Third party involvement	Model generalizability
Data publishing	Arora and Upadhyay (2019) Wu et al. (2022), etc.	data	no	no	no
Federated learning	Xie et al. (2021), etc.	model gradients	yes	yes	yes
Privacy-preserving network embedding	Wang et al. (2021) Han et al. (2024), etc.	embeddings	yes	no	no
Ours		**model**	**yes**	**no**	**yes**

Appendix B. Training Algorithm and Complexity

Algorithm 1 presents the detailed training process for the privacy-preserving pre-trained GNN model as outlined in Sections 4.2 to 4.4, which primarily consists of five parts: graph data augmentation, privacy-preserving operator, GNN model, contrastive learning and generalizability learning.

Algorithm 1 Training algorithm.

Input: The training graph of data owner G_{train}, set of data augmentation functions T, and set of diverse data augmentation functions Q.
Output: The pre-trained GNN model e_θ.

for sampled minibatch of data $\{G_n : G_n \in G_{\text{train}}\}$ **do**
  for $n = 1$ to $|V_{\text{train}}|$ **do**
  Part I: Graph data augmentation
    Sample graph data augmentation functions t_i, t_j from T
    Generate first augmented view: $t_i(G_n)$
    Generate second augmented view: $t_j(G_n)$
  Part II: Privacy-preserving operator
    Generate $t_i(G_n)$'s edge-removal candidate set $\mathcal{R}_{t_i(G_n)}$ and edge-keeping candidate set $\mathcal{K}_{t_i(G_n)}$
    Generate first privacy-preserving augmented view $G_{n,p\circ t_i} = (p \circ t_i)(G_n)$, by removing edges in $\mathcal{R}_{t_i(G_n)} \setminus \mathcal{K}_{t_i(G_n)}$ from $t_i(G_n)$.
    Generate $t_j(G_n)$'s edge-removal candidate set $\mathcal{R}_{t_j(G_n)}$ and edge-keeping candidate set $\mathcal{K}_{t_j(G_n)}$
    Generate second privacy-preserving augmented view $G_{n,p\circ t_j} = (p \circ t_j)(G_n)$, by removing edges in $\mathcal{R}_{t_j(G_n)} \setminus \mathcal{K}_{t_j(G_n)}$ from $t_j(G_n)$.
  Part III: GNN model
    Compute representation: $e_\theta(G_{n,p\circ t_i})$
    Compute representation: $e_\theta(G_{n,p\circ t_j})$
  end for
  Part IV: Contrastive learning
  Define $l_{\text{cl},n} = -\log \frac{\exp(\text{sim}(e_\theta(G_{n,p\circ t_i}), e_\theta(G_{n,p\circ t_j}))/\tau)}{\sum_{n' \neq n} \exp(\text{sim}(e_\theta(G_{n,p\circ t_i}), e_\theta(G_{n',p\circ t_i}))/\tau)}$, where τ is the temperature hyper-parameter
  $\mathcal{L}_{\text{pretrain}} = \frac{1}{|V_{\text{train}}|} \sum_{n=1}^{|V_{\text{train}}|} l_{\text{cl},n}$
  Part V: Generalizability learning
  Generate K diverse graph data augmentation functions from Q, forming $\mathcal{Q} = \{q_1, q_2, \cdots q_K\}$
  $\mathcal{L}_{\text{general}} = \frac{1}{|V_{\text{train}}|} \sum_{n=1}^{|V_{\text{train}}|} \text{VAR}(\{\|e(G_{n,p\circ q_i}) - \mu\| : q_i \in \mathcal{Q}\})$
  Update the GNN model e_θ to minimize $\mathcal{L}_{\text{pretrain}} + \beta \mathcal{L}_{\text{general}}$
end for

The time complexity of our model mainly consists of five modules: graph data augmentation, privacy-preserving operator, GNN model propagation, contrastive learning, and generalizability learning. Suppose the maximal number of nodes and number of edges of subgraph instances is $|V_{\text{sub}}|$ and $|E_{\text{sub}}|$, the batch size is B, and D is the representation dimension. (1) As for the graph data augmentation, the time complexity of random walk with restart is at least $O(B|V_{\text{sub}}|^3)$ (Xia et al. 2019) and the heuristic measures is $O(B|V_{\text{sub}}|^2)$. (2) The time complexity of privacy-preserving operator is $O(|E_{\text{sub}}|)$. (3) The time complexity of GNN encoder propagation depends on the architectures of the backbone GNN. We use GIN (Xu et al. 2019) as our GNN encoder, whose complexity is $O(B|E_{\text{sub}}|)$ here. (4) The time complexity of the contrastive loss is $O(B^2D)$ (Li et al. 2022). (5) The time complexity of the generalizability learning is $O(KBD)$. Therefore, the overall time complexity of our model in each batch is $O(B|V_{\text{sub}}|^3 + |E_{\text{sub}}| + B|E_{\text{sub}}| + B^2D + KBD)$. Given that K is a much smaller constant relative to B, this complexity simplifies to $O(B|V_{\text{sub}}|^3 + B|E_{\text{sub}}| + B^2D)$, making it comparable to that of the basic graph pre-training benchmark model GCC, which is $O(B|V_{\text{sub}}|^3 + B|E_{\text{sub}}| + B^2D)$. It is important to note that our model is scalable to large-scale training graph, as its complexity depends only on the maximal number of nodes and edges of ego networks in the training graph, where the size of ego networks would not be excessively large.

Appendix C. Experimental Details

Additional datasets and scenarios. We also explore scenarios where the data owner's and model user's graphs differ in relationship types and are from distinct temporal snapshots, namely cross-relation and cross-time respectively.

In the cross-relation scenario, the data owner and model user possess graphs with different types of relations. For the evaluation in this scenario, we processed the Twitter dataset (Sabuncu 2020) with different re-tweet relations. The Twitter dataset is generated based on 24M US election-related tweets spanning from July 1 to November 11, 2020. Nodes are users and edges represent their re-tweet relations. Each node's label indicates the sentiment of the tweet, i.e., positive or negative. The re-tweet relations stand for the political party to which the tweet is related. In our constructed scenario, the data owner processes a re-tweet graph by creating an edge between users if the tweet contains any keyword related to the Democratic party. Another re-tweet graph is established by edges between users if the tweet does not contain any keywords related to the Democratic or Republican parties. To ensure the data owner's data advantage, the model user's downstream graph is constructed by sampling only half of the nodes from this graph. The edges among these nodes are then formed by conducting random walks starting from each node (Lovász and Winkler 1995).

Table C1 Dataset statistics.

Dataset	Graph	# Nodes	# Edges	# Class	Dataset	Graph	# Nodes	# Edges	# Class
Deezer	G_{train}	28,281	214,308	2	Amazon	G_{train}	34,755	25,170	4
	G_{down}	14,140	52,706			G_{down}	19,860	7,268	
	$G_{\text{train}} \cap G_{\text{down}}$	14,140	36,538			$G_{\text{train}} \cap G_{\text{down}}$	11,962	6,871	
	$G_{\text{train}} \cup G_{\text{down}}$	28,281	230,476			$G_{\text{train}} \cup G_{\text{down}}$	42,653	25,567	
Facebook-Page	G_{train}	22,470	364,358	4	Twitter	G_{train}	48,406	44,873	2
	G_{down}	11,235	91,524			G_{down}	21,514	10,554	
	$G_{\text{train}} \cap G_{\text{down}}$	11,235	54,352			$G_{\text{train}} \cap G_{\text{down}}$	13,119	6,072	
	$G_{\text{train}} \cup G_{\text{down}}$	22,470	401,530			$G_{\text{train}} \cup G_{\text{down}}$	56,801	49,355	
LastFM	G_{train}	7,624	63,196	18	Twitter-Foursquare	G_{train}	5,120	130,575	-
	G_{down}	3,812	15,550			G_{down}	2,656	11,289	
	$G_{\text{train}} \cap G_{\text{down}}$	3,812	10,454			$G_{\text{train}} \cap G_{\text{down}}$	903	2,412	
	$G_{\text{train}} \cup G_{\text{down}}$	7,624	68,292			$G_{\text{train}} \cup G_{\text{down}}$	6,873	139,452	
DBLP	G_{train}	8,485	21,666	10	Phone-Email	G_{train}	1,000	41,191	-
	G_{down}	4,186	5,182			G_{down}	501	1,239	
	$G_{\text{train}} \cap G_{\text{down}}$	2,035	1,092			$G_{\text{train}} \cap G_{\text{down}}$	500	705	
	$G_{\text{train}} \cup G_{\text{down}}$	10,636	25,756			$G_{\text{train}} \cup G_{\text{down}}$	1,001	41,725	

Notes: G_{train} and G_{down} are the graphs of data owner and model user, respectively.

In the cross-time scenario, the data owner and model user possess graphs with different time snapshots. For the evaluation in this scenario, we processed two datasets, DBLP (Lu et al. 2019), Amazon (Leskovec et al. 2007), with edges in different time snapshots. Specifically, DBLP is a co-author temporal network with 27 time snapshots, in which nodes are authors and an edge between two nodes exists if the authors have collaborated on at least one paper together. The node labels of DBLP indicate the 10 research areas of the authors. In our constructed scenario of DBLP, the data owner and model user process co-author networks in the first and the second time snapshots respectively. Amazon is a co-purchasing temporal network with four time snapshots "March 2", "March 12", "May 5", "June 1", in which nodes are products and edges are their co-purchasing relations. The node labels of Amazon are the product groups like book, music, video and more. In our constructed scenario using the DBLP dataset, the data owner processes co-purchasing network at the time snapshots of "March 2". Additionally, another co-purchasing network from the time snapshot of "March 12" is extracted. For the model user's downstream graph, it is derived by removing half of the nodes randomly from the "March 12" network. The edges among the remaining nodes are then sampled through random walks starting from each node (Lovász and Winkler 1995).

The statistics of all the datasets used are listed in Table C1.

Details of baselines. We compare with the following benchmarks:

- GAL-W (Liao et al. 2021) is an adversarial learning based method that filters out private information via adversarial representation learning based on Wasserstein distance.
- GAL-TV (Liao et al. 2021) is also an adversarial learning based method that is similar to GAL-W but replaces the Wasserstein distance with the Total Variation distance.

- EdgeRand (Wu et al. 2022) is a differential privacy based methods, which imposes data perturbation by randomly flipping each entry in the adjacency matrix of the training graph. The perturbed graph is then input to our graph pre-training model.
- LapGraph (Wu et al. 2022) is also a differential privacy based method that adds Laplace noise to the adjacency matrix while better preserving the density of the training graph. The obtained perturbed graph is also input to our graph pre-training model.
- GCC (Qiu et al. 2020) is a state-of-the-art graph pre-training model with generalizability. It does not take the privacy leakage into consideration.

Implementation details. We conduct all experiments on a single machine of Linux system with an AMD EPYC-Rome and NVIDIA RTX 3090 GPU (24G memory). All models are implemented in PyTorch[1] version 1.12.1 with CUDA version 11.3 and Python 3.8.

When implementing our method, during pre-training, we use Adam for optimization (β_1=0.9, β_2=0.999, and ϵ=1×10^{-8}. The weight decay was set to 1e-4. The gradient clipping was set to 1.0, and the dropout was set to 0.5. When conducting node classification, link prediction, and adversarial link predictor, we use the Adam optimizer with learning rate of 0.01 for 50 epochs.

The hyper-parameter settings of all baselines in the evaluation phase were the same as ours while those in the training phase were tuned to their optimal values. When implementing GCC (Qiu et al. 2020), the total iteration number was set to 132, the batch size was set to 128 and learning rate was 5e-3. The number of graph encoder layers was set to 5 and the hidden size of each layer was set to 64. The gradient clipping was set to 1.0, and the dropout was set to 0.5. When implementing GAL-W and GAL-TV (Liao et al. 2021), the learning rate of adversarial trainers was set to 1e-3. The attackers were trained for 30 epochs with a batch-size of 128 nodes before the original model was trained for 120 epochs after 35 epochs of pre-training, with a batch-size of 128 nodes. The number of graph encoder layers was set to 5, and the hidden size of each layer was set to 64. The trade-off hyper-parameter λ between the accuracy and information leakage was set to 0.8. When implementing EdgeRand and LapGraph (Wu et al. 2022), the number of graph encoder layers was set to 4 and the hidden size of each layer was set to 64. The dropout rate was set to 0.5. The training epoch was set to 200 with 5e-3 learning rate. The privacy-utility trade-off hyper-parameter ϵ was set to 5. All baseline models were trained with Adam optimizer (β_1=0.9, β_2=0.999, and ϵ=1×10^{-8}) (Kingma and Ba 2014).

Appendix D. Additional Experiments

[1] https://github.com/pytorch/pytorch.

Table D1 **The privacy-preserving performance of private link reconstruction attack, reported as AUC (Std).**

Model	Private link reconstruction attack (AUC%)		
	DBLP	Amazon	Twitter
GCC	75.99 (0.42)	78.73 (0.19)	72.79 (0.22)
GAL-W	65.67 (0.08)	68.80 (0.43)	65.17 (0.64)
GAL-TV	68.32 (0.17)	71.03 (0.40)	66.39 (0.61)
EdgeRand	73.82 (0.20)	76.32 (0.12)	71.15 (0.27)
LapGraph	72.55 (0.15)	75.56 (0.59)	70.23 (0.19)
Ours	69.92 (0.19)	72.77 (0.43)	67.58 (0.29)

Table D2 **Generalizability performance on node classification and link prediction, reported as Micro F1 (Std).**

Model	Node classification (Micro F1%)			Link prediction (Micro F1%)		
	DBLP	Amazon	Twitter	DBLP	Amazon	Twitter
GCC	25.84 (0.20)	40.76 (0.29)	66.01 (0.31)	87.37 (0.26)	85.89 (0.44)	74.53 (0.41)
GAL-W	14.16 (0.09)	33.37 (0.28)	55.87 (0.47)	73.78 (0.48)	70.77 (0.27)	64.72 (0.26)
GAL-TV	16.84 (0.39)	36.28 (0.12)	57.21 (0.63)	74.41 (0.35)	73.15 (0.74)	64.11 (0.25)
EdgeRand	24.57 (0.31)	40.52 (0.41)	64.54 (0.46)	86.88 (0.69)	86.09 (0.55)	74.37 (0.44)
LapGraph	25.06 (0.64)	39.86 (0.50)	64.30 (0.40)	87.18 (0.85)	86.82 (0.56)	75.02 (0.73)
Ours	25.93 (0.40)	40.98 (0.31)	65.37 (0.15)	87.43 (0.41)	87.21 (0.41)	75.40 (0.30)

Experiments on cross-relation and cross-time scenarios. Table D1 and Table D2 show the capabilities of various methods in terms of privacy preservation and generalizability across cross-relation and cross-time scenarios. While methods like EdgeRand, LadGraph, and GCC show good generalizability, they lack in privacy preservation. Conversely, GAL-W and GAL-TV, despite their privacy-preserving attributes, fall short on generalizability, limiting their use in model-sharing contexts. Our model strikes a better balance between generalizability and privacy preservation.

Ablation studies. We conduct ablation studies to show the effectiveness of each component in our privacy-preserving operator and the generalizability learning. Specifically, we conduct ablation studies comparing the performance of our model, our model excluding generalizability score in the privacy-preserving operator (denoted as Ours w/o p_{xy}^{keep}), our model that replaces selective removal of edges based on p_{xy}^{del} with random edge removal in the privacy-preserving operator (denoted as Ours w/o p_{xy}^{del}), our model entirely without the privacy-preserving operator (denoted as Ours w/o $p(\cdot)$), and the one without the generalizability learning (denoted as Ours w/o general). The results in Table D3 show that: (1) Our model without considering p_{xy}^{keep} would enhance privacy but at a significant cost to generalizability. This underscores the critical need for including p_{xy}^{keep} in the design. (2) Removing p_{xy}^{del} substantially compromises the model's privacy-preserving capabilities, which demonstrates the necessity of including p_{xy}^{del} in the design. (3) Comparing our model to the one without the privacy-preserving operator (Ours w/o $p(\cdot)$), it is evident that the operator significantly boosts privacy protection without detrimentally affecting generalizability. This demonstrates that the

Table D3 Ablation studies.

Model	Private link reconstruction attack (AUC%) ↓						Downstream performance on node classification (Micro F1%) ↑					
	Deezer	Facebook-Page	LastFM	DBLP	Amazon	Twitter	Deezer	Facebook-Page	LastFM	DBLP	Amazon	Twitter
Ours	82.35	83.80	83.55	69.92	72.77	67.58	56.05	49.62	23.57	25.93	40.98	65.37
Ours w/o p_{xy}^{keep}	79.37	81.29	80.39	66.73	69.84	66.02	53.39	47.95	20.89	23.75	39.28	63.29
Ours w/o p_{xy}^{del}	86.79	86.26	89.47	76.92	79.15	73.95	56.74	49.98	23.69	26.35	41.26	65.98
Ours w/o $p(\cdot)$	87.95	87.70	88.82	77.49	80.41	74.38	57.32	50.25	23.62	26.08	41.54	66.15
Ours w/o general	82.04	83.46	84.04	70.33	72.54	68.29	55.40	48.90	23.45	25.29	40.72	65.12

design of the privacy-preserving operator effectively enhances the trade-off. (4) The generalizability learning enhances generalizability without compromising its privacy-preserving capabilities. This underscores that our design of generalizability learning can achieve a superior trade-off.

Training time comparison. We further report the training time comparison between our model and the baselines on the Deezer dataset. The training time of GCC, GAL-W, GAL-TV, EdgeRand, LapGraph and our method are 3.65, 591.25, 477.81, 14.39, 18.72 and 5.79 seconds per epoch. We can seethat the training time of our model is significantly shorter than that of most other baselines. The only exception is the GCC, a graph pre-training method without privacy preservation; however, whereas it can be trained with the shortest time, it consistently exhibits the poorest privacy-preserving performance (refer to Table 1 in the main text). Adversarial training methods GAL-W and GAL-TV are heavily constrained by the substantial computational costs during model training due to the iterative and interactive training processes. Considering both training time and model performance, our model exhibits a compelling advantage in achieving a favorable balance between privacy preservation and model generalizability, all while maintaining low computational costs.

We also note that the time complexity of benchmarks EdgeRand and LapGraph (Wu et al. 2022) consist of two components: (1) DP mechanism, whose time complexity is $O(|V_{\text{train}}|^2)$, where $|V_{\text{train}}|$ is the number of nodes in G_{train}. (2) GNN encoder, which is GCN (Kipf and Welling 2017) with time complexity of $O(|E_{\text{train}}|)$, where $|E_{\text{train}}|$ is the number of edges in G_{train}. Consequently, the overall time complexity of EdgeRand and LapGraph is $O(|V_{\text{train}}|^2 + |E_{\text{train}}|)$. This indicates that these methods are less scalable to large-scale training graphs due to their dependency on the number of nodes and edges in the training graph. Additionally, the benchmarks GAL-W and GAL-TV (Liao et al. 2021), which utilize adversarial training methods, significantly increase the runtime compared to their non-adversarial counterparts (Wong et al. 2020). Our experimental results for training time comparison also show that GAL-W and GAL-TV require approximately 102 and 83 times the training time of our model, respectively. Moreover, they utilize GCN (Kipf and Welling 2017) as the GNN encoder, whose time complexity is $O(|E_{\text{train}}|)$. This further underscores their lack of scalability to large-scale training graph.

Table D4 The effect of graph properties on privacy leakage risk.

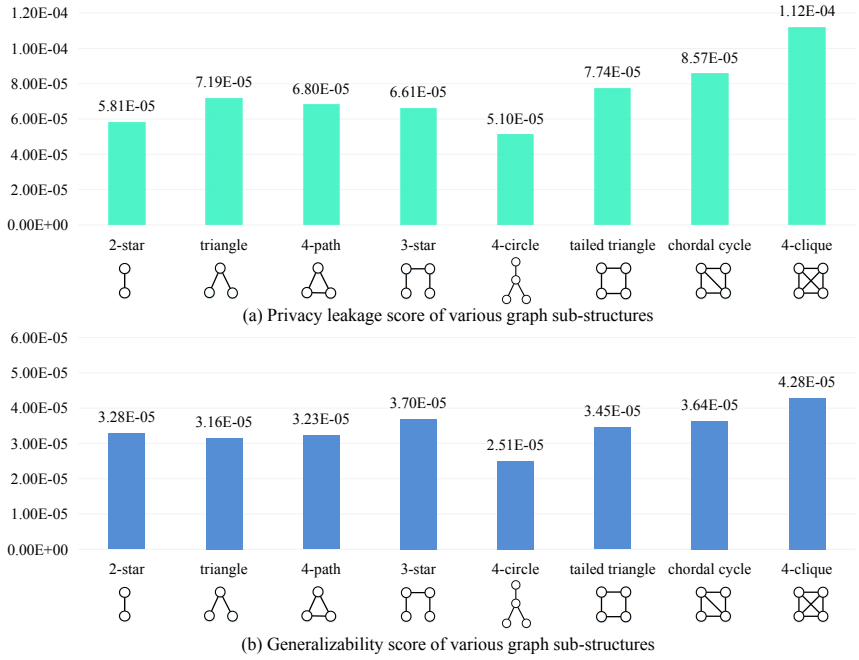
Dataset	$\|V_{\text{train}}\|$	$\|E_{\text{train}}\|$	average degree	degree variance	graph density	network entropy	clustering coefficient	transitivity	network constraint	scale-free exponent	degree centrality	eigenvector centrality	**privacy protection**
Deezer	28281	214308	6.56	63.14	5.36E-04	3.99	0.1412	0.09592	0.41998	5.03	2.32E-04	1.17E-03	**4.5%**
Facebook-Page	22470	364358	15.22	697.69	1.44E-03	5.13	0.3597	0.23232	0.31780	3.24	6.77E-04	1.32E-03	**8.9%**
LastFM	7624	63196	7.29	132.23	2.18E-03	4.03	0.2194	0.17862	0.44297	3.33	9.57E-04	2.00E-03	**5.8%**

Notes: We report the average value of some node-level properties including network constraint, degree centrality and eigenvector centrality.

Case studies. Building upon the proposed model and baselines, we provide additional insights through case studies that empirically explore the relationship between graph properties, sub-structures, and the capabilities of privacy preservation and model generalizability.

Case 1: How graph properties affect privacy-preserving capability? We first aim to investigate which graph properties exhibited by a graph dataset are more likely to leak privacy. We compare the privacy-preserving capability (as seen in the last column of Table D4) with the graph properties (as seen in the columns of Table D4 except for the last one) of Deezer, Facebook-Page, and LastFM datasets. Specifically, we include a variety of graph-level and node-level properties: average degree, degree variance, graph density, network entropy (Gómez-Gardenes and Latora 2008), clustering coefficient, transitivity, network constraint (Burt 2004), scale-free exponent, degree centrality, and eigenvector centrality (Bonacich 1987). The privacy-preserving capability is quantified by measuring how much our model reduces the performance of private link reconstruction attack compared with the graph pre-training model without privacy protection (i.e., GCC). For example, on Deezer dataset, our model reduces 4.5% (= $|82.35 - 86.19|/86.19$; 82.35 and 86.19 are the performance of our model and GCC under private link reconstruction attack, reported in Table 1 of the main text) performance of private link reconstruction attack compared with GCC.

The results show that we could significantly reduce the privacy risk on Facebook-Page dataset. On the other hand, it is more difficult to protect the privacy on the Deezer and LastFM datasets, whose properties show a trend of the high value of average degree, degree variance, network entropy, clustering coefficient, and transitivity, and low value of network constraint and scale-free exponent, which are less complex in general. The high values of average degree, degree variance, network entropy, clustering coefficient, and transitivity indicate that the network is highly connected and has a large number of interdependent nodes. Nodes in highly connected networks should have a greater influence on each other, increasing privacy breach risks by allowing one compromised node to impact many others quickly (Chen et al. 2012). Additionally, tracking and controlling the flow of sensitive information becomes more challenging due to the higher number of pathways for data flow (Venkatesh et al. 2020). On the other hand, datasets characterized by low network

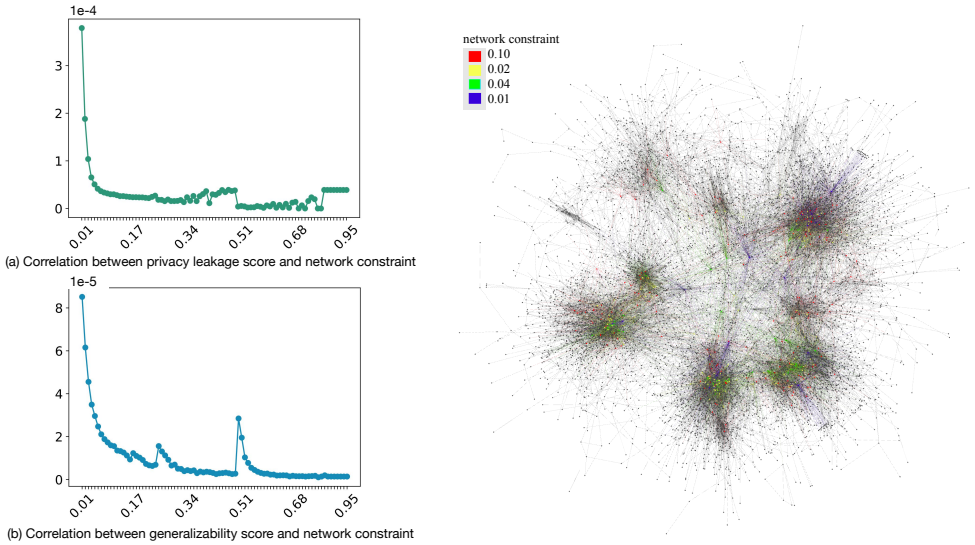


Notes: A higher privacy leakage score (p_{xy}^{del}) indicates the substructure has a higher risk of leaking privacy. A higher generalizability score (p_{xy}^{keep}) indicates the substructure is more semantic-related (i.e., more important or influential) in the original graph.

Figure D1 The average privacy leakage and generalizability scores for different graph sub-structures.

constraint and scale-free exponent are less complex and have fewer inter-node connections, making them vulnerable to privacy breaches. Individual nodes in such datasets have a greater impact on the network structure, increasing the risk of privacy exposure.

Case 2: How graph sub-structures and structural holes affect generalizability and privacy-preserving capability? We further analyze how different graph sub-structures and structural holes affect privacy preservation and model generalizability. We take the LastFM dataset as an illustration as it demonstrates properties that lie between those of the three real-world social network datasets Deezer, Facebook-Page and LastFM (as shown in Table D4). We explore typical sub-structures such as 2-star, triangle, 4-path, 3-star, 4-circle, tailed triangle, chordal cycle, and 4-clique. We also consider structural holes by measuring the network constraint of a node. The lower the network constraint of a node is, the more likely the node is to act as a gatekeeper (i.e., network bridge or broker) between distinct groups or clusters of nodes in the graph (Burt 2004). Utilizing privacy leakage scores (p_{xy}^{del}) and generalizability scores (p_{xy}^{keep}) derived from our graph data augmentation process, we evaluate the privacy preservation and model generalizability implications Higher privacy leakage scores suggest weaker privacy-preserving capabilities, whereas higher generalizability scores indicate better model generalizability. This dual assessment helps in understanding



Notes: Different colors indicate the nodes with different value of network constraint.

Figure D2 Analysis of network constraint of nodes on LastFM graph.

the trade-offs between privacy preservation and generalizability. We conduct the analyze on graph sub-structures and structural holes as follows.

- Sub-structures: Figure D1 presents the privacy leakage score and the generalizability score for different graph sub-structures. The privacy leakage and generalizability scores of each sub-structure are calculated by averaging all edges' p_{xy}^{del} and p_{xy}^{keep}. We have two observations: (1) The results show that sub-structures like tailed triangle, chordal cycle, and 4-clique have higher privacy leakage scores, exceeding the average by 4.9%, 16.2%, and 51.8%, respectively, showing lower privacy-preserving capability. Simultaneously, their generalizability scores are also higher than average by 1.3%, 6.9%, and 25.7%, respectively, suggesting better generalizability. Therefore, balancing privacy preservation and generalizability is crucial when handling graph sub-structures like tailed triangles, chordal cycles, and 4-cliques. (2) We also observe that, for the 3-star graph sub-structure, although it has a high generalizability score compared to others, its privacy leakage score is lower than most of other sub-structures. A plausible reason is that 3-star sub-structure usually has fewer internal connections and may be more likely to disconnect from the rest of the network, which can make it easier to protect privacy. This indicates that data owners can prioritize generalizability by preserving semantic meanings in 3-star sub-structure, given the sub-structure's robust privacy-preserving capability. On the contrary, for the triangle graph sub-structure, although it has a lower generalizability score than most of other sub-structures, its privacy leakage score is relatively high. In the case of dealing with triangle, data owners should prioritize privacy protection, which is more important, since the graph data have fewer semantic meanings preserved but are more likely to leak privacy.
- Structural holes: Figure D2 presents the correlation between privacy/generalizability score and the measure of the structural hole, i.e., network constraint. The privacy leakage score and generalizability score of each node are calculated as the mean value of all edges' p_{xy}^{del} and p_{xy}^{keep} connected with the node, respectively. We posit that nodes with lower network constraints, i.e., often serving as gatekeepers, would be more likely to leak private information while maintaining generalizability. This is possible because these nodes often hold critical positions in the network's

connectivity and structure (Newman 2003). Compromising or manipulating these nodes can significantly affect the privacy of other connected nodes.

References

Arora R, Upadhyay J (2019) On differentially private graph sparsification and applications. *Advances in neural information processing systems* 32.

Bonacich P (1987) Power and centrality: A family of measures. *American journal of sociology* 92(5):1170–1182.

Burt RS (2004) Structural holes and good ideas. *American journal of sociology* 110(2):349–399.

Chen D, Lü L, Shang MS, Zhang YC, Zhou T (2012) Identifying influential nodes in complex networks. *Physica a: Statistical mechanics and its applications* 391(4):1777–1787.

Gómez-Gardenes J, Latora V (2008) Entropy rate of diffusion processes on complex networks. *Physical Review E* 78(6):065102.

Han X, Yang Y, Wang L, Wu J (2024) Privacy-preserving network embedding against private link inference attacks. *IEEE Transactions on Dependable and Secure Computing* 21(2):847–859.

Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. *arXiv:1412.6980* .

Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*.

Leskovec J, Adamic LA, Huberman BA (2007) The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)* 1(1):5–es.

Li S, Wang X, Zhang A, Wu Y, He X, Chua TS (2022) Let invariant rationale discovery inspire graph contrastive learning. *International Conference on Machine Learning*, 13052–13065 (PMLR).

Liao P, Zhao H, Xu K, Jaakkola T, Gordon GJ, Jegelka S, Salakhutdinov R (2021) Information obfuscation of graph neural networks. *International Conference on Machine Learning*, 6600–6610 (PMLR).

Lovász L, Winkler P (1995) Mixing of random walks and other diffusions on a graph. *London Mathematical Society Lecture Note Series* 119–154.

Lu Y, Wang X, Shi C, Yu PS, Ye Y (2019) Temporal network embedding with micro- and macro-dynamics. *Proceedings of the 28th ACM international conference on information and knowledge management*, 469–478.

Newman ME (2003) The structure and function of complex networks. *SIAM review* 45(2):167–256.

Qiu J, Chen Q, Dong Y, Zhang J, Yang H, Ding M, Wang K, Tang J (2020) Gcc: Graph contrastive coding for graph neural network pre-training. *SIGKDD*, 1150–1160.

Sabuncu I (2020) Usa nov. 2020 election 20 mil. tweets (with sentiment and party name labels) dataset. *URL https://dx. doi. org/10.21227/25te-j338* .

Venkatesh P, Dutta S, Grover P (2020) Information flow in computational systems. *IEEE Transactions on Information Theory* 66(9):5456–5491.

Wang B, Guo J, Li A, Chen Y, Li H (2021) Privacy-preserving representation learning on graphs: A mutual information perspective. *KDD*, 1667–1676.

Wong E, Rice L, Kolter JZ (2020) Fast is better than free: Revisiting adversarial training. *International Conference on Learning Representations*.

Wu F, Long Y, Zhang C, Li B (2022) Linkteller: Recovering private edges from graph neural networks via influence analysis. *IEEE Symposium on Security and Privacy (S&P'22)*.

Xia F, Liu J, Nie H, Fu Y, Wan L, Kong X (2019) Random walks: A review of algorithms and applications. *IEEE Transactions on Emerging Topics in Computational Intelligence* 4(2):95–107.

Xie H, Ma J, Xiong L, Yang C (2021) Federated graph classification over non-iid graphs. *Advances in neural information processing systems* 34:18839–18852.

Xu K, Hu W, Leskovec J, Jegelka S (2019) How powerful are graph neural networks? *International Conference on Learning Representations*.