TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 03/18 pp244–256
DOI: 10.26599/TST.2021.9010015
Volume 27, Number 2, April 2022

Link-Privacy Preserving Graph Embedding Data Publication with Adversarial Learning

Kainan Zhang, Zhi Tian, Zhipeng Cai, and Daehee Seo*

Abstract: The inefficient utilization of ubiquitous graph data with combinatorial structures necessitates graph embedding methods, aiming at learning a continuous vector space for the graph, which is amenable to be adopted in traditional machine learning algorithms in favor of vector representations. Graph embedding methods build an important bridge between social network analysis and data analytics, as social networks naturally generate an unprecedented volume of graph data continuously. Publishing social network data not only brings benefit for public health, disaster response, commercial promotion, and many other applications, but also gives birth to threats that jeopardize each individual's privacy and security. Unfortunately, most existing works in publishing social graph embedding data only focus on preserving social graph structure with less attention paid to the privacy issues inherited from social networks. To be specific, attackers can infer the presence of a sensitive relationship between two individuals by training a predictive model with the exposed social network embedding. In this paper, we propose a novel link-privacy preserved graph embedding framework using adversarial learning, which can reduce adversary's prediction accuracy on sensitive links, while persevering sufficient non-sensitive information, such as graph topology and node attributes in graph embedding. Extensive experiments are conducted to evaluate the proposed framework using ground truth social network datasets.

Key words: graph embedding; privacy preservation; adversarial learning

1 Introduction

Over the last decade, social networks have evolved from being an entertaining extra to an integrated part of nearly every aspect of people's daily life. The shipment of social media users in January 2020 was about 3.8 billion, with 7 percent increasing rate per year. Social networks not only make it convenient for users to interact with their friends and family members, but also enable third parties to utilize published social network data for various purposes, such as promoting business, improving healthcare, preventing pandemic and disasters, etc.[1–5], taking advantage of powerful data mining and machine learning techniques. On the other hand, combinatorial structures of graph data limit the widespread adoption of most machine learning methods, which only support vector representations as an input. Graph embedding methods have emerged as a promising solution to transform graph data to a set of low-dimensional vectors, such that a richer set of statistics and machine learning approaches can be utilized[6].

Note that the set of low-dimensional representation vectors need to preserve the graph topology and other relevant information about graphs. Therefore, publishing graph embedding data has to face the same dilemma as

• Kainan Zhang and Zhipeng Cai are with the Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA. E-mail: kzhang16@student.gsu.edu; zcai@gsu.edu.
• Zhi Tian is with the Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA 22030, USA. E-mail: ztian1@gmu.edu.
• Daehee Seo is with the National Center of Excellence in Software, Sangmyung University, Seoul 03016, Republic of Korea. E-mail: daehseo@smu.ac.kr.
∗ To whom correspondence should be addressed.
Manuscript received: 2020-12-26; revised: 2021-02-07; accepted: 2021-02-23

publishing original social network data, both of which suffer from threatening user privacy seriously. Although we can sanitize a social network by modifying the original social graph structure in a precise way to retain utility and preserve user privacy[7], implementing graph embedding directly after traditional data anonymization may not perform well, because this decoupled behavior limits the expressiveness of the model. Furthermore, as most influential graph embedding methods concern both local neighborhood structure and global graph topology, substantial information about the edges of a removed node/user can still be retrieved from the embedding vectors of the remaining adjacent nodes[8]. Recently, differential privacy is introduced to graph embedding, which intends to make no statistical difference on the output of the released graph embedding matrix after the removal or addition of an edge in the original graph. However, the sensitivity of multiple-dimensional graph data is so high, such that it may lead to poor utility with extra computation cost. Therefore, it is worth to discover a balanced trade-off between sharing social graph embedding and compromising user privacy.

In this paper, we address a practical issue named celebrity privacy, which refers to the right of celebrities and public figures to withhold the information that they are unwilling to disclose to the public[9]. Different from the privacy of the general public, celebrity privacy is usually challenged by the press for commercial purposes or the fans for personal interests. Moreover, because a celebrity usually has a wider social circle, it is easier to predict more private information through inference attacks. Do Chinese basketball player Yao Ming and England football player David Beckham know each other? The answer is an easy yes, as long as we know that both of them were invited by Britain's Prince William to attend a Wild-Aid campaign[10]. This privacy issue can be generalized as how to preserve relationship information among some users who have extensive and strong relationships in a social network graph. Although simply deleting a relationship link between two users can make the first-order relationship irrelevant, high-order information (i.e., common friends and attributes) will force the embedding of these sensitive users to be closely related to each other and adversaries can still predict the relationship with high accuracy through machine learning techniques. Privacy customization is another challenge for this issue that the celebrity or data owner may only want to protect partial relationship-links. To overcome the aforementioned problems and reduce the risk concern about privacy disclosure in social network data publishing, we propose a Link-Privacy Preserving Graph Embedding (LPPGE) framework using adversarial learning with two types of prepossessing methods. The main contributions of our work can be summarized as follows:

- We investigate a practical privacy issue named celebrity privacy based on some graph embedding methods, which are widely used for social network analysis, and our proposed work can protect the specified links upon user requirements.
- Our framework integrates both graph embedding and social network privacy protection into an end-to-end process flow through an adversarial training based graph autoencoder.
- Extensive experiments on ground truth social network data demonstrate the performance of our framework in privacy protection compared with other existing methods.

2 Related Work

2.1 Graph embedding methods

As the first unsupervised graph representation learning model, DeepWalk[11] traverses a network via random walk for sampled vertex sequences leveraging skip-gram[12] to learn the latent vertex representation. Based on the architecture of DeepWalk, Node2Vec[13] optimizes the random walk strategy for vertex sampling by discovering both structural equivalence and homophily of the graph. LINE[14] defines two objective functions (i.e., first-order proximity and second-order proximity) to preserve both the local and global network structures by minimizing the difference between the input and embedding distributions. It is suitable for directed and/or weighted information networks. As an extension of LINE, SDNE[15] uses an autoencoder to optimize two proximity functions simultaneously and is robust to sparse networks. Not only considering graph structures, GAE[16] uses Graph Convolutional Networks (GCNs) to encode nodes in a graph and improves performance by incorporating node features. Further works[17, 18] prove that GAEs can be successfully used for link prediction and recommendation systems. Although these prior graph embedding methods have been making good progress in different fields, less consideration is given to the privacy issue in graph embedding publishing.

2.2 Link privacy preservation in social networks

With the investigation of Online Social Networks (OSNs) diving deeper, more concerns about users' privacy have been rising. Korolova et al.[19] showed that attackers can piece together the link structure of a global network by only bribing several users and their neighborhood information. Ying and Wu[20] investigated the performance of edge randomization on protecting the privacy of sensitive links and showed that attackers can enhance the link prediction accuracy via node proximity measures. Fire et al.[21] reconstructed the removed links by training a link prediction classifier with a small amount of training data. Fard and Wang[22] proposed a structure-aware randomization scheme that hides a sensitive link with less distortion in a directed graph. Considering the dependence between utility/public attributes, private/public attributes, and link information, Cai et al.[23] developed a collective method to sanitize social networks against inference attacks on user profile and friendship relations. While the above works have achieved some decent trade-offs between data utility and privacy, none of them integrates its technology with graph embedding which is an effective and efficient way in graph analysis. Close to our work, DPNE[24] is the first work on preserving differential privacy in network embedding, and PPGD[25] develops a differentially private perturbed gradient descent method for matrix-factorization based graph embedding matrix sharing. Although differential privacy is considered as the gold standard of rigorous privacy, it cannot guarantee the protective effect against the aforementioned problems. Moreover, the performances of DPNE and PPGD in graph representation are also limited by the matrix factorization based methods.

3 Problem Statement

We firstly introduce the definitions to state our problem as follows.

Social network: We model a social network as an undirected graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{X})$, consisting of a set of users $\boldsymbol{V}$, friendship-link set $\boldsymbol{E}$, and user-attribute set $\boldsymbol{X}$. $\boldsymbol{A}$ is the adjacency matrix corresponding to the structure of graph $\boldsymbol{G}$. If $\boldsymbol{e}(i, j) \in \boldsymbol{E}$ (i.e., users u_i and u_j are friends), then $\boldsymbol{A}_{ij} = 1$, otherwise $\boldsymbol{A}_{ij} = 0$.

As we are interested in protecting the celebrity users, the sensitive user is defined as follows.

Sensitive user: Given a social network $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{X})$ and λ, if user $u_i \in \boldsymbol{V}$ and the node degree of $u_i \geqslant \lambda$, u_i is a sensitive user. The value of λ is pre-defined by the data owner, which is usually larger than the average degree of graph $\boldsymbol{G}$.

Sensitive link: Any pair of sensitive users u_i and u_j in social network $\boldsymbol{G}$, where $\boldsymbol{e}(i, j) \in \boldsymbol{E}$ is a candidate sensitive link. The candidate sensitive link set is defined as CS. The data owner can customize the set of actual sensitive links $S \subseteq CS$. Links in $\boldsymbol{E} \setminus CS$ are considered as non-sensitive links.

Graph embedding is generally used for a variety of machine learning tasks, such as node classification and link prediction. The ultimate goal of our method is to publish a link-privacy preserving graph embedding vectors without sacrificing data utility/usability. In the following, we define data utility and link-privacy in our LPPGE.

Privacy: We define privacy as the prediction accuracy for the set of sensitive links S by a classifier C_1, which is trained by graph embedding to predict links in a social network graph.

Utility: We define utility as the amount of information to be preserved in the graph embedding from an original graph $\boldsymbol{G}$, which is measured by the non-sensitive link classification accuracy using classifier C_1, and the node classification accuracy using classifier C_2.

Thus, our proposed method is expected to derive a privacy-preserving graph embedding, that can achieve the desired privacy-utility trade-off between privacy and utility.

4 Proposed Framework

We design the LPPGE framework based on Graph Auto-Encoder (GAE)[16], shown as Fig. 1, which makes use of graph structure $\boldsymbol{A}$ and node content $\boldsymbol{X}$ to learn a latent representation $\boldsymbol{Z}$, and then reconstructs $\hat{\boldsymbol{A}}$ from $\boldsymbol{Z}$. In the probabilistic setting of GAE, the encoder is defined by a posterior $q(\boldsymbol{Z}|\boldsymbol{A}, \boldsymbol{X})$, where H^i is the matrix of i-th hidden layer, and the decoder is defined by a generative distribution $p(\boldsymbol{A}|\boldsymbol{Z})$.

We also utilize a supervised learning mechanism in

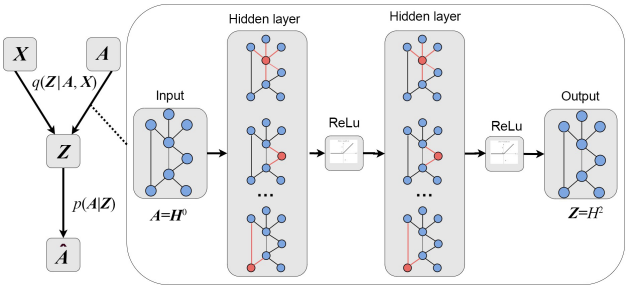

Fig. 1 GAE with graph convolutional networks.

Adversarial Auto-Encoder (AAE)[26] to achieve privacy protection. Classical AAE forces the latent code to match the previous distribution through the adversarial training module, which distinguishes whether the current latent code $z_i \in \boldsymbol{Z}$ comes from the encoder or the previous distribution. While in supervised AAE, shown as Fig. 2, a label vector z_p is provided to the decoder along with the latent code z_i to reconstruct the information. The encoder must disentangle some information from z_i to make z_i obey the previous distribution, while the decoder can gain the label information from $\boldsymbol{Z_p}$, so that the label information can be disentangled from z_i during the reconstruction.

4.1 Prepossessing

As shown in Fig. 3, there are two prepossessing steps before applying the adversarial learning scheme.

First, we delete all the sensitive links in the original graph $\boldsymbol{G}$ to obtain a modified graph $\boldsymbol{G}_{train}$ as the input of the training process to avoid computing sensitive link prediction loss during reconstruction, which can remove the first-order sensitive link information from the graph embedding.

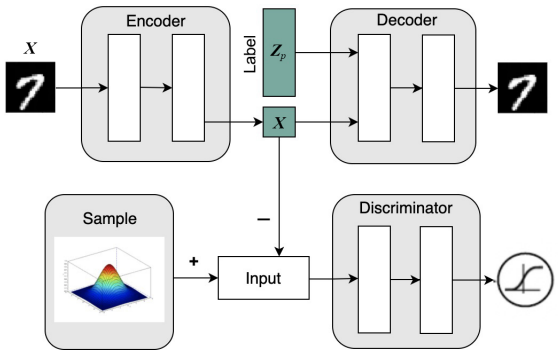

Fig. 2 Supervised AAE.

Second, a privacy embedding is generated via another privacy graph $\boldsymbol{G}_{priv}$. Because we want to exclude the private information in the graph embedding, a privacy label that represents that information explicitly should be provided to the decoder. However, each embedding vector is a representation of each node, not each link, then how can we add a privacy label on each node to include the link information is a challenge. Here, different from the independent one-hot label information in supervised AAE, our private information is located in a pair of nodes' embedding vectors (i.e., a link is determined by two nodes' labels). Note that the private information is not only the direct sensitive link between sensitive users, but also includes the high-order information of the sensitive users (e.g., mutual friends and user profiles), which can be used to infer the first-order friendship. Therefore, instead of deleting all the non-sensitive links in $\boldsymbol{G}$, we develop two methods to generate the privacy graph $\boldsymbol{G}_{priv}$ separately:

(1) Trimming method: We define a radius R, which is the minimum graph eccentricity of a sensitive user as the central point, and only keep the links within R in $\boldsymbol{G}_{priv}$ (e.g., when $R = 1$, only the links that connect sensitive users will be kept; when $R = 2$, only the links that connect sensitive users or their neighbors will be

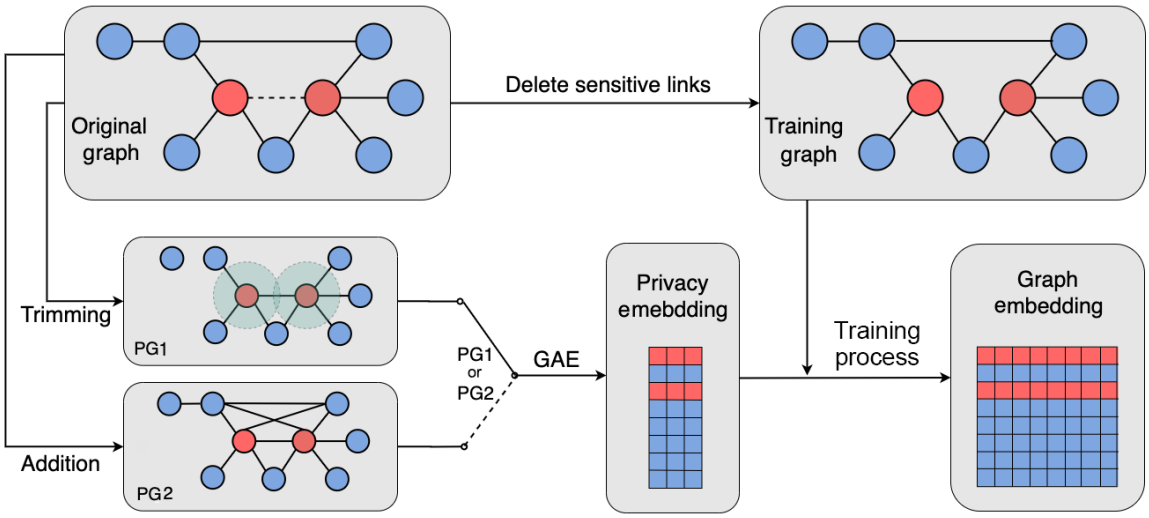

Fig. 3 Graph prepossessing. The red nodes are sensitive users ($\lambda = 4$) and the blue ones are non-sensitive users. In the original graph, the dotted line is the sensitive link which is deleted in the training graph. In the Privacy Graph 1 (PG1), two links out of radius ($R = 1$) are deleted. In the Privacy Graph 2 (PG2), two links are added between sensitive and non-sensitive users. The privacy embedding will be generated from either privacy graph.

kept).

(2) Addition method: To enhance relationship between sensitive users, we add M links in $\boldsymbol{G}$ by Algorithm 1. In Step 2 of Algorithm 1, We set an upper-bound U for the number of links to be distributed near each sensitive link $\mathbf{se}(i,j)$ based on the node degrees of its associated sensitive users u_i and u_j. Because we consider higher degree users as more sensitive, a stronger relationship needs to be built between them. In Steps 3 and 4 of Algorithm 1, we connect u_j with u_i's friends conditionally, so that more private information can be embedded into the privacy graph embedding framework.

At last, we compute the privacy graph embedding $\boldsymbol{Z_p}$ of $\boldsymbol{G}_{priv}$ generated from either method, and use it as a privacy label which describes the relationship of sensitive users precisely.

4.2 Encoder model

Based on the design of GAE and AAE, LPPGE consists of an encoder, a decoder, and a discriminator, which is shown in Fig. 4. The encoder involves GCNs which extends the operation of convolution to graph data in the spectral domain, and learns a layer-wise transformation by spectral convolution function,

$$\boldsymbol{Z}^{(l+1)} = f(\boldsymbol{Z}^{(l)}, \boldsymbol{A}|\boldsymbol{W}^{(l)}) \tag{1}$$

where $\boldsymbol{Z}^{(l)}$ is the input for convolution, $\boldsymbol{Z}^{(l+1)}$ is the output after convolution, and $\boldsymbol{W}^{(l)}$ is the weight matrix

Algorithm 1 Generate privacy graph by addition method

Input: $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{X})$: original graph
$\mathbf{SE}$: sensitive links in $\boldsymbol{G}$
N: number of $\mathbf{SE}$
M: number of links to be added
D: average node degree of sensitive users
$d(i)$: node degree of user $u_i \in \boldsymbol{V}$
$\mathbf{se}(i,j)$: sensitive link between users u_i and u_j
Output: privacy graph $\boldsymbol{G}_{priv}$
1: **for** $\mathbf{se}(i,j) \in \mathbf{SE}$ **do**
2: Compute the upper-bound, $U = \frac{M}{N} \times \frac{d(i)+d(j)}{2D}$
3: **if** $\exists\, u_i$'s friend u_x who is 2 hops away from u_j without passing u_i **then**
4: Link u_x to u_j
if the number of new links added for $\mathbf{se}(i,j) \leqslant U$ then
find next u_x
else
break
end if-else
5: **end if**
6: **end for**
7: **return** the modified $\boldsymbol{G}$ as the privacy graph $\boldsymbol{G}_{priv}$

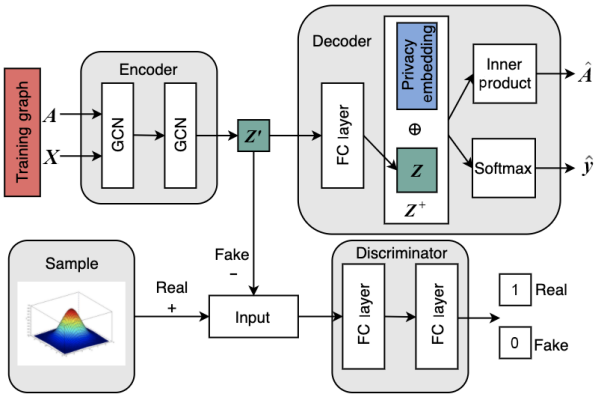

Fig. 4 Architecture of the adversarial learning in LPPGE.

of l-th layer to be learned during training.

We have $\boldsymbol{Z^0} = \boldsymbol{X} \in \mathbb{R}^{n\times m}$ (n nodes and m features) for our problem. The symmetrically normalized graph Laplacian is applied in $f(\cdot)$ as

$$f(\boldsymbol{Z}^{(l)}, \boldsymbol{A}|\boldsymbol{W}^{(l)}) = \phi(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{Z}^{(l)}\boldsymbol{W}^{(l)}) \tag{2}$$

where $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ ($\boldsymbol{I}$ is the identity matrix of $\boldsymbol{A}$) and $\tilde{\boldsymbol{D}}_{ii} = \sum_j \tilde{\boldsymbol{A}}_{ij}$ (i.e., the diagonal node degree matrix of $\tilde{\boldsymbol{A}}$), and ϕ is an activation function.

In LPPGE, we use a 2-layer GCN to extract latent code $\boldsymbol{Z}'$ from graph $\boldsymbol{G}_{train}$. The activation function of the first layer is $Relu(\cdot)$ and the second layer is a linear function. In order to disentangle the private information from latent code $\boldsymbol{Z}'$, we apply a privacy embedding on $\boldsymbol{Z}'$ before feeding it to the decoder. In this way, the encoder learns a compressed $\boldsymbol{Z}'$, which excludes the private information, but is sufficient for the decoder to reconstruct the graph data because of merging privacy label encoding. Although a lower-dimensional $\boldsymbol{Z}'$ can extrude more private information, it will also lose some utility information to reconstruct the graph. To moderate the performance decrements, we use the method presented by Li et al.[27], which mapped the low dimensional representation $\boldsymbol{Z}'$ to a higher dimensional $\boldsymbol{Z}$ via a fully connected layer to restore the utility information. Then we can concatenate the privacy embedding with $\boldsymbol{Z}$ to obtain $\boldsymbol{Z^+}$ as the input of the decoder.

4.3 Decoder model

Because the final embedding result should be able to reconstruct $\boldsymbol{G}_{train}$ with adjacency matrix (structure) $\boldsymbol{A}$ and content information $\boldsymbol{X}$, there are two modules in the decoder. The first module reconstructs an adjacency matrix $\hat{\boldsymbol{A}}$ via the inner product of embedding matrix

$\boldsymbol{Z}^+$ as

$$\hat{\boldsymbol{A}} = \sigma((\boldsymbol{Z}^+)(\boldsymbol{Z}^+)^{\mathrm{T}}) \tag{3}$$

where $\sigma(\cdot)$ is the logistic sigmoid function.

The cross-entropy loss between $\hat{\boldsymbol{A}}$ and $\boldsymbol{A}$ would be minimized as the loss for link prediction,

$$\mathcal{L}_{link} = -\frac{1}{N^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\boldsymbol{A}_{ij}\log(\hat{\boldsymbol{A}}_{ij}) \tag{4}$$

where $\boldsymbol{A}_{ij}$ and $\hat{\boldsymbol{A}}_{ij}$ are the corresponding elements of $\boldsymbol{A}$ and $\hat{\boldsymbol{A}}$, respectively.

The second module is a category classifier, which decodes $\boldsymbol{Z}^+$ using a soft-max function $\hat{\boldsymbol{y}}_i = \mathrm{softmax}(z_i^+)$ and computes the cross-entropy loss between the one-hot label $\boldsymbol{y}_i$ of each user. The loss function is defined as

$$\mathcal{L}_{label} = -\frac{1}{N}\sum_{i=1}^{n}\boldsymbol{y}_i\log(\hat{\boldsymbol{y}}_i) \tag{5}$$

Thus the total loss of LPPGE is the combination of the link prediction loss and the category classification loss,

$$\mathcal{L}_{recon} = \mathcal{L}_{link} + \alpha\mathcal{L}_{label} \tag{6}$$

where α is a trade-off parameter between the link prediction loss and the category classification loss.

4.4 Discriminator model

The discriminator consists of two fully connected layers. It will be trained to distinguish whether a latent code is from Gaussian distribution (positive) or from the encoder of LPPGE (negative). We optimize the discriminator by minimizing the following loss:

$$\mathcal{L}_{dc} = -\log(D(\boldsymbol{x})) - \log(1 - D(\boldsymbol{z}_i')) \tag{7}$$

where $D(\boldsymbol{x})$ is the discriminator's estimation of the probability that real sample $\boldsymbol{x}$ is from the Gaussian distribution, and $D(\boldsymbol{z}_i')$ is the discriminator's estimation of the probability that a latent code $\boldsymbol{z}_i'$ is real. During the optimization, the graph embedding $\boldsymbol{Z}$ is regularized to Gaussian distribution.

4.5 Algorithm explanation

We summarize the proposed framework in Algorithm 2. The framework has two stages: prepossessing (Steps 1 and 2) and training (Steps 3 to 10). In Step 8, we update the discriminator to tell if the input is from a positive sample or the graph encoder. In Step 9, the encoder is updated to confuse the discriminator. These two steps can be integrated as $\min\max\mathcal{L}_{dc}$ in Eq. (7). After T epochs' training, we return the final link-privacy preserved graph embedding $\boldsymbol{Z} \in \mathbb{R}^{n\times d}$ in Step 11.

Algorithm 2 Link-privacy preserving graph embedding

Input: $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{X})$: original graph
T: number of training iterations
d: dimension of the final graph embedding $\boldsymbol{Z}$
Output: $\boldsymbol{Z} \in \mathbb{R}^{n\times d}$

1: Generate the training graph $\boldsymbol{G}_{train}$ and the privacy graph $\boldsymbol{G}_{priv}$ from $\boldsymbol{G}$
2: Generate the privacy embedding $\boldsymbol{Z}_p$ from $\boldsymbol{G}_{train}$
3: **for** *iteration* $= 0$ to T **do**
4: Generate the latent code $\boldsymbol{Z}'$ of $\boldsymbol{G}_{train}$ by Eq. (2)
5: Map $\boldsymbol{Z}'$ to the higher dimensional $\boldsymbol{Z}$
6: Concatenate $\boldsymbol{Z}_p$ with $\boldsymbol{Z}$ as the input of the decoder
7: Update the encoder and decoder by minimizing Eq. (6)
8: Update the discriminator by minimizing Eq. (7)
9: Update the encoder by maximizing Eq. (7)
10: **end for**
11: **return** $\boldsymbol{Z} \in \mathbb{R}^{n\times d}$

5 Experiment

In our experiments, we employ four different datasets summarized in Table 1. The first two datasets are **Cora** and **Citeseer** used in Ref. [16], and both of them consist of scientific publications as nodes and citation relationships as edges. The features are unique words in each document. The other two datasets are two ground truth social network datasets: **Yale** and **Rochester**, which are composed of Facebook users from Yale University and Rochester University.

By default, we assume the number of the sensitive links is around 1% of the total links, then we find the corresponding λ to define the sensitive users and sensitive links in each dataset. For the addition method, we set the target number M to be 10% of the total links and the actual number is around 9%.

We compare LPPGE with the following baselines:

Table 1 The employed graph datasets and the parameters of each dataset. Per* is the percentage of the sensitive links in the graph. The actual number of added_Links (addition method) is less than M because of the graph structure and upper-bound U round-down for each sensitive link.

Dataset	#Nodes	#Links	#Features	λ	#Sen_Links	Per* (%)	Radius R	α	M	#Added_Links
Cora	2708	5429	1433	13	53	0.97	2	3	470	434
Citeseer	3327	4732	3703	16	62	1.31	1	3	40 000	39 200
Yale	8758	405 450	7	320	2800	0.69	2	2	40 000	39 200
Rochester	4563	167 653	7	220	1446	0.86	2	2	16 000	15 906

(1) GAE is an autoencoder-based model for unsupervised learning on graph-structured data.

(2) GAE_RM uses the same framework as GAE, but deletes the defined sensitive links.

(3) DPNE is a differentially private network embedding method based on DeepWalk as matrix factorization. It applies the objective perturbation approach by adding noise in the objective function of matrix factorization to learn a representation satisfying differential privacy.

(4) Non-Privacy Graph Embedding (NPGE) uses the same architecture as LPPGE, but the privacy embedding of NPGE is generated from the graph that only has sensitive links.

For LPPGE, we implement **LPPGE(T)** using the trimming method to generate the privacy graph and **LPPGE(A)** using the addition method. We set the embedding size of DPNE as 64 dimensions with a relatively large privacy budget $\epsilon = 1$ to maximize the defined utility for all the datasets. For the rest of the methods, we embed a graph into a 16-dimensional space for the Cora and Citeseer datasets, while a 32-dimensional space for the Yale and Rochester datasets. In LPPGE, we also set hidden code z' to be a 4-dimensional vector for all the datasets. The performance of LPPGE is evaluated on two aspects which are link prediction and node classification.

5.1 Link prediction

We train a Multi-Layer Perceptron (MLP) classifier as the attacker, which tries to predict the sensitive links in a social network by the graph embedding. The input of the MLP is the embedding vectors of two users in the social network and the output is the relationship between these two users. We assume 10% of non-sensitive links are exposed to the attacker as the positive samples in the training set, and the same number of negative samples are collected by randomly selecting unconnected users. We present the results in the Macro F1-score of the non-sensitive links and the sensitive links separately. We conduct each experiment 10 times and show the mean values with standard errors as the final scores. The details of the experiment results on link prediction are shown in Table 2.

The performance of GAE_RM shows that even if the sensitive links are deleted, the attacker is still able to predict its existence precisely according to the high-order information. Although DPNE has the lowest prediction accuracy for sensitive links, its prediction accuracy of non-sensitive links is much lower than other methods. It can be seen DPNE costs a significant amount of utility information in its embedding to preserve privacy, and differential privacy is susceptible to inference attacks. The result of NPGE shows that merely extruding sensitive links in graph embedding is not enough to protect sensitive information. Moreover, it should be noted that as sensitive users have large degrees, in the traditional graph embedding methods, sensitive links are easier to be predicted than non-sensitive links, which can be observed in Table 2. It is also worth mentioning that both LPPGE(T) and LPPGE(A) can reduce sensitive accuracy much lower than non-sensitive ones on the Cora and Citeseer datasets. For the Yale and Rochester datasets, LPPGE(A) can reduce sensitive accuracy by about 10% with only losing 5% utility accuracy on non-sensitive links. These facts prove that LPPGE is capable of reducing the attackers' prediction accuracy of sensitive links, while slightly sacrificing the utility of embedding to reconstruct the non-sensitive part of a graph.

5.2 Node classification

We apply two classifiers, MLP and Support Vector Machine (SVM), to predict user category labels through graph embedding. For all the datasets, 5-fold cross-validation is used to ensure the model's reliability and

Table 2 Result of link prediction.

(%)

Approach	Dataset							
	Cora		Citeseer		Yale		Rochester	
	Non-sensitive	Sensitive	Non-sensitive	Sensitive	Non-sensitive	Sensitive	Non-sensitive	Sensitive
GAE	81.8 ± 1.0	85.6 ± 3.5	83.7 ± 1.4	92.1 ± 3.4	84.4 ± 1.4	90.0 ± 1.2	85.1 ± 0.4	90.7 ± 1.1
GAE_RM	84.6 ± 0.7	83.3 ± 3.0	87.5 ± 1.2	91.8 ± 3.3	84.2 ± 0.2	89.6 ± 0.8	84.7 ± 0.3	88.5 ± 0.7
DPNE	55.3 ± 1.3	67.3 ± 4.2	52.7 ± 1.3	67.5 ± 5.2	48.8 ± 10	52.5 ± 10	60.6 ± 1.2	75.9 ± 3.9
NPGE	85.1 ± 1.4	89.2 ± 2.8	88.9 ± 1.5	92.6 ± 2.6	81.7 ± 0.1	85.2 ± 0.6	83.5 ± 0.4	88.0 ± 1.2
LPPGE(T)	81.5 ± 1.1	75.8 ± 3.9	85.6 ± 1.6	81.0 ± 3.9	81.2 ± 0.2	83.1 ± 0.5	82.2 ± 0.3	83.1 ± 0.7
LPPGE(A)	**80.5 ± 1.2**	**72.3 ± 3.1**	**84.1 ± 1.2**	**77.0 ± 3.5**	**80.2 ± 0.2**	**80.7 ± 1.0**	**80.4 ± 0.3**	**81.3 ± 1.2**

effectiveness, and the results are given as Macro F1-score in Table 3.

Because DPNE only embeds graph structure information without node attributes in the embedding, even though with a higher dimensional embedding space, the performance is still poor on the classification task and the graph embedding utility is low. Comparing LPPGE with GAE and GAE_RM, the classifiers have similar accuracy in predicting node's class labels on all the datasets, which indicates LPPGE can maintain accurate cluster information at the same level with privacy protection.

5.3 Trade-off between utility and privacy

To demonstrate the trade-off between utility and privacy, we compute the ratio of utility (i.e., the sum of the prediction accuracy of non-sensitive links and node classification) to privacy (i.e., the prediction accuracy of sensitive links), that is Utility/Priracy. As shown in Fig. 5, both LPPGE(T) and LPPGE(A) can achieve better performance on the aspects of privacy protection and data usability preservation. Due to space limitation, the experiment results for link prediction and node classification are presented in the Appendix with Table A1 listing all notations used in the paper.

5.4 Graph visualization

We visualize the Cora and Yale datasets in 2-dimensional space by applying the t-SNE algorithm to the learned embedding. The Cora dataset is partitioned by the publication subject, and the Yale dataset is partitioned by the user's class year. Each subgroup is represented in a different color. The results shown in Figs. 6 and 7 validate our assertions in the node classification section through a meaningful layout.

Table 3 Result of node classification. Dataset(∗) indicates the number of the categories in each dataset.

(%)

Approach	Cora(7)		Citeseer(6)		Yale(6)		Rochester(2)	
	MLP	SVM	MLP	SVM	MLP	SVM	MLP	SVM
GAE	74.0	71.5	58.6	54.0	85.8	87.0	84.8	84.1
GAE_RM	75.1	71.3	63.5	55.5	85.1	86.3	85.8	84.4
DPNE	14.5	6.63	18.6	8.32	24.2	4.66	50.2	44.7
NPGE	72.4	68.0	69.0	64.1	78.9	77.6	84.1	80.0
LPPGE(T)	79.2	70.8	56.4	47.7	84.8	83.5	86.2	82.8
LPPGE(A)	73.9	71.8	66.9	62.3	84.0	83.3	86.4	83.6

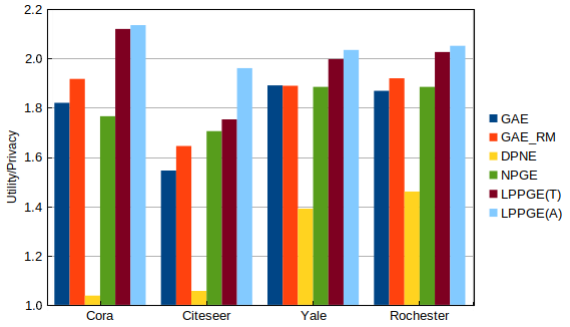

Fig. 5 Evaluation of trade-off between utility and privacy.

5.5 Customization

In real-world situations, data owners may only need to protect some specified relationships between sensitive users based on their demands. Hence, we randomly select 25%, 50%, and 75% links from pre-defined sensitive links as the new sensitive links and conduct the same Utility/Privacy evaluation to testify the scalability of our model. The results for different scales shown in Fig. 8 demonstrate that LPPGE can fulfill customized requests for privacy protection. We present the link prediction and node classification results of different customized datasets in Tables A2–A9 in the Appendix.

6 Conclusion and Future Work

In this paper, we investigate a practical privacy issue in social graph embedding and propose a novel graph embedding framework. By utilizing graph autoencoder and adversarial learning, we can regularize the latent representation to follow a prior distribution and extrude sensitive information from graph embedding. The experiment results demonstrate that our framework can achieve the desired trade-off between privacy protection and data usability comparing with other non-privacy preserving methods. In the future, we would like to further explore more applications for dynamic graph embedding as a new trending research direction. Adversaries may predict or restore sensitive information by capturing the dynamism of graph sequence[28], which is a big challenge in data privacy and security.

Acknowledgment

This work was supported by the National Science Foundation of USA (Nos. 1829674, 1912753, 1704287, and 2011845). The authors would like to thank the referees for their valuable comments and helpful suggestions.

References

[1] J. S. He, M. Han, S. Ji, T. Du, and Z. Li, Spreading social influence with both positive and negative opinions in online networks, *Big Data Mining and Analytics*, vol. 2, no. 2, pp. 100–117, 2019.

[2] X. Zheng, G. C. Luo, and Z. P. Cai, A fair mechanism for private data publication in online social networks, *IEEE*

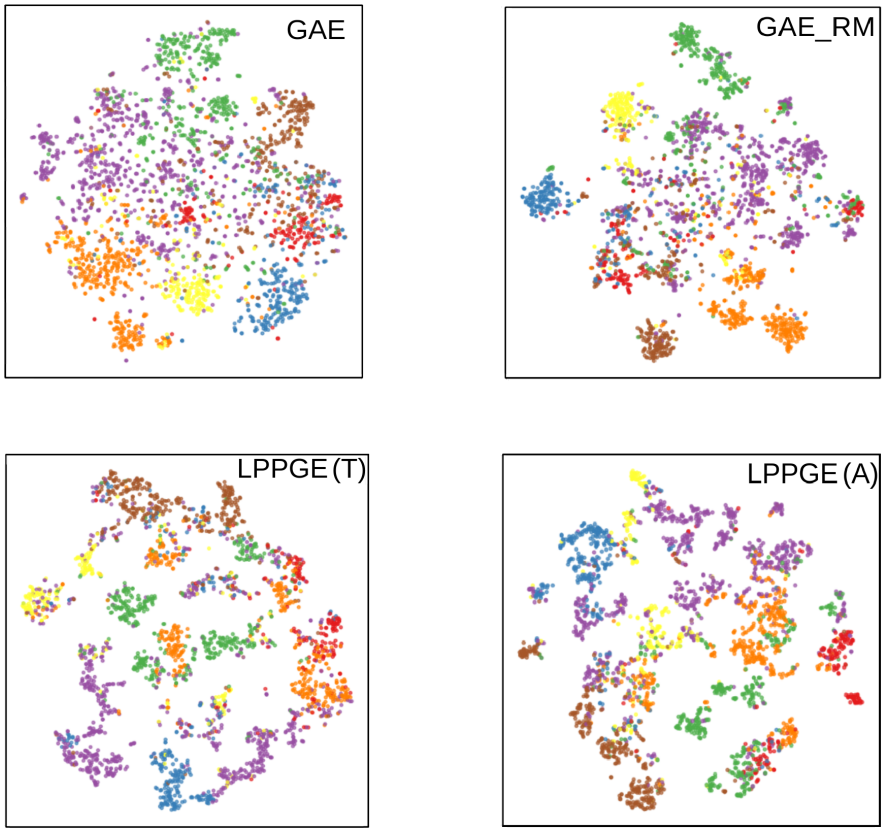

Fig. 6 Visualization comparison for the Cora dataset.

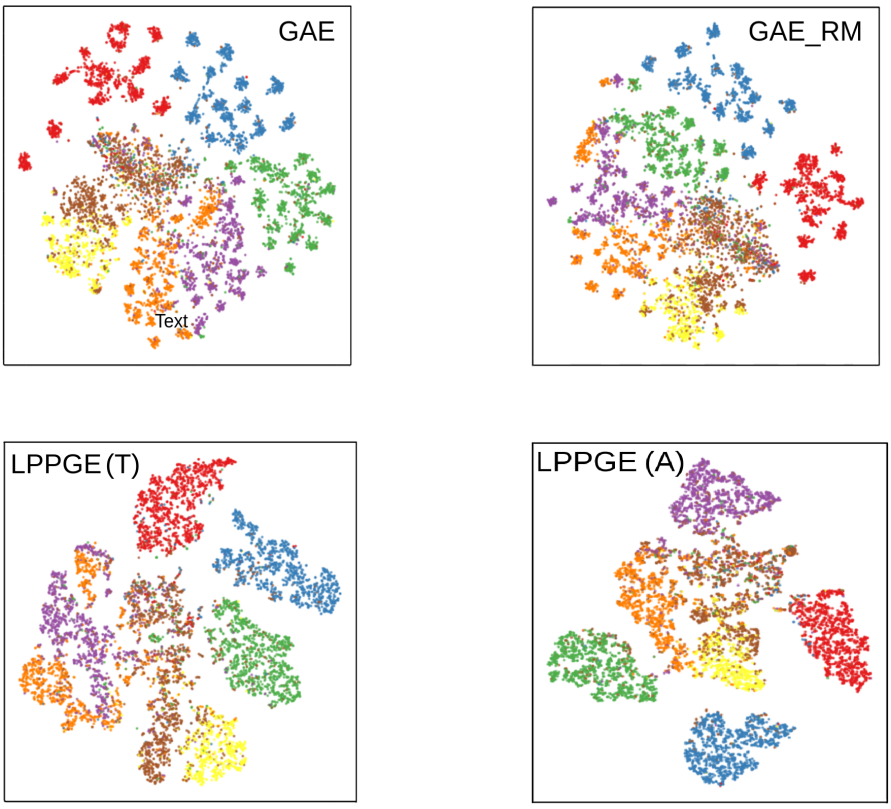

Fig. 7 Visualization comparison for the Yale dataset.

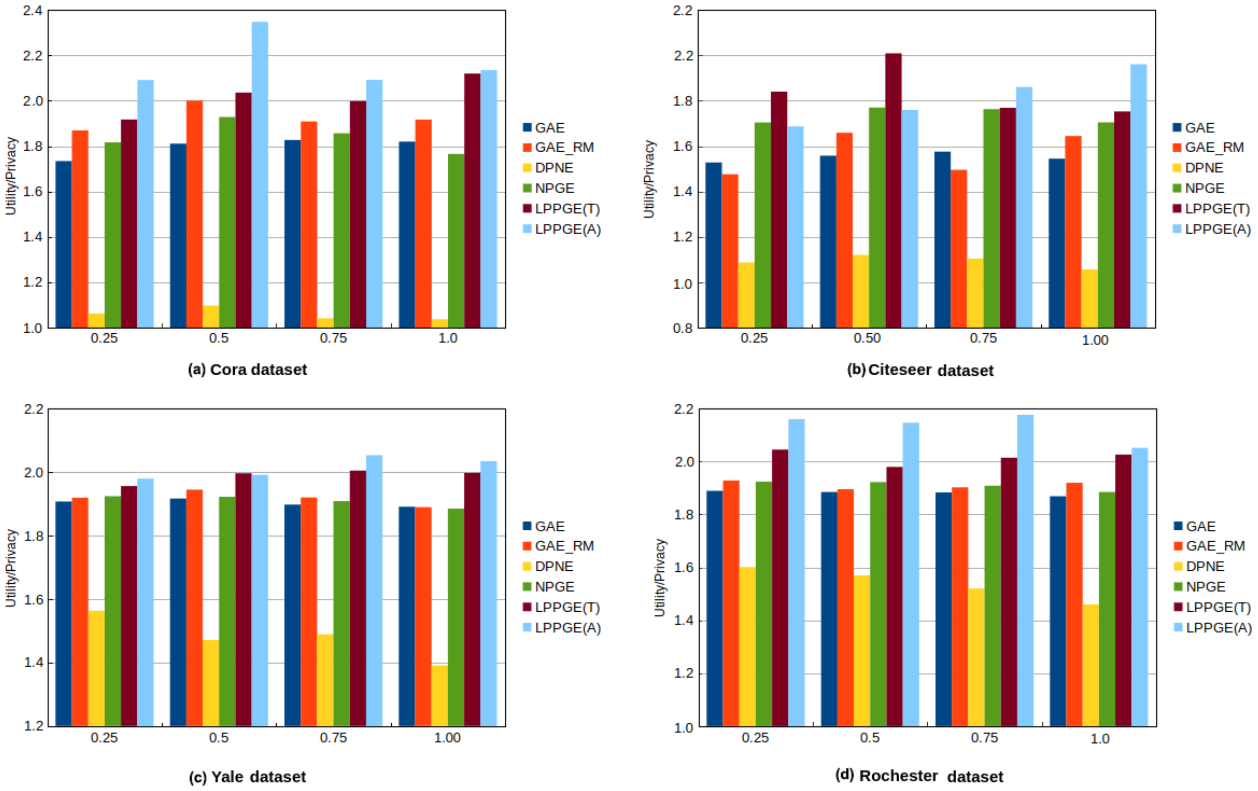

Fig. 8 Evaluation of trade-off between utility and privacy for different scales. *X*-axis is the percentage of sensitive links.

Appendix

Table A1 List of notations used in this paper.

Symbol	Description	Symbol	Description
$\boldsymbol{G}$	Undirected social network graph	$\boldsymbol{Z_p}$	Label information provided to decoder
$\boldsymbol{V}$	User set	z_p	Latent code(vector) $\in \boldsymbol{Z_p}$
$\boldsymbol{E}$	Friendship-link set	$\boldsymbol{G}_{train}$	Training graph
$\boldsymbol{X}$	User-attribute set	$\boldsymbol{G}_{priv}$	Privacy graph
$\boldsymbol{A}$	Adjacency matrix corresponding to the graph structure	R	Radius defined in trimming method
u_i	User $i \in \boldsymbol{V}$	M	Number of links added in addition method
$\boldsymbol{e}(i,j)$	Friendship between users u_i and u_j	U	Upper-bound of links to be distributed near each sensitive link
λ	Lower bound of sensitive user's node degree		
S	Candidate sensitive link	$\mathbb{R}^{n\times m}$	n nodes and m features
CS	Candidate sensitive link set	$\boldsymbol{I}$	Identity matrix
$E \setminus CS$	Non-sensitive links	$\tilde{\boldsymbol{D}}$	Diagonal node degree matrix
$\mathbf{se}(i,j)$	Sensitive link between users u_i and u_j	ϕ	Activation function
C_1	Link prediction classifier	$\boldsymbol{Z}'$	Initial latent representation learned by LPPGE
C_2	Node classification classifier	$\boldsymbol{Z}^+$	$\boldsymbol{Z}$ Concatenated with the privacy embedding
$\boldsymbol{Z}$	Latent representation learned from $\boldsymbol{A}$ and $\boldsymbol{X}$	$\boldsymbol{y}$	One-hot label of user
$\hat{\boldsymbol{A}}$	Graph structure reconstructed by $\boldsymbol{Z}$	α	Trade-off parameter between the link prediction loss and the category classification loss
z_i	Latent code(vector) $\in \boldsymbol{Z}$		

Transactions on Network Science and Engineering, vol. 7, no. 2, pp. 880–891, 2020.

[3] M. Siddula, Y. S. Li, X. Z. Cheng, Z. Tian, and Z. P. Cai, Anonymization in online social networks based on enhanced Equi-Cardinal clustering, *IEEE Transactions on Computational Social Systems*, vol. 6, no. 4, pp. 809–820, 2019.

[4] X. Zheng, Z. P. Cai, G. C. Luo, L. Tian, and X. Bai, Privacy-preserved community discovery in online social networks, *Future Generation Computer Systems*, vol. 93, pp. 1002–

Table A2 Result of link prediction for Cora dataset.

(%)

Approach	25%		50%		75%		100%	
	Non-sensitive	Sensitive	Non-sensitive	Sensitive	Non-sensitive	Sensitive	Non-sensitive	Sensitive
GAE	82.2 ± 1.7	89.9 ± 4.6	82.0 ± 1.2	86.3 ± 3.4	82.4 ± 1.8	85.1 ± 4.0	81.8 ± 1.0	85.6 ± 3.5
GAE_RM	80.8 ± 1.0	82.1 ± 6.9	84.5 ± 1.3	81.3 ± 4.8	85.0 ± 1.1	84.5 ± 4.1	84.6 ± 0.7	83.3 ± 3.0
DPNE	54.6 ± 1.3	65.7 ± 6.1	55.0 ± 1.7	63.7 ± 8.1	53.8 ± 1.7	65.9 ± 4.3	55.3 ± 1.3	67.3 ± 4.2
NPGE	86.4 ± 1.4	85.7 ± 5.2	86.3 ± 1.0	82.6 ± 3.9	84.3 ± 2.4	85.2 ± 3.4	85.1 ± 1.4	89.2 ± 2.8
LPPGE(T)	81.9 ± 1.8	81.1 ± 4.3	82.4 ± 1.1	77.7 ± 3.9	82.4 ± 1.3	78.1 ± 3.2	81.5 ± 1.1	75.8 ± 3.9
LPPGE(A)	**80.7 ± 1.0**	**73.0 ± 5.1**	**80.7 ± 0.9**	**64.6 ± 4.7**	**81.2 ± 1.2**	**72.6 ± 3.4**	**80.5 ± 1.2**	**72.3 ± 3.1**

Note: 25%, 50%, 75%, and 100% indicate the percentage of pre-defined sensitive links for customization, same as in Tables A3–A9.

Table A3 Result of link prediction for Citeseer dataset.

(%)

Approach	25%		50%		75%		100%	
	Non-sensitive	Sensitive	Non-sensitive	Sensitive	Non-sensitive	Sensitive	Non-sensitive	Sensitive
GAE	84.9 ± 1.2	94.3 ± 3.2	84.8 ± 2.6	93.7 ± 2.1	85.0 ± 1.7	91.3 ± 3.6	83.7 ± 1.4	92.1 ± 3.4
GAE_RM	85.0 ± 1.6	97.0 ± 4.4	86.0 ± 1.8	87.2 ± 6.0	86.6 ± 0.9	92.6 ± 2.6	87.5 ± 1.2	91.8 ± 3.3
DPNE	52.9 ± 1.7	64.0 ± 6.4	52.2 ± 1.5	62.2 ± 6.0	53.4 ± 1.6	64.0 ± 4.4	52.7 ± 1.3	67.5 ± 5.2
NPGE	87.5 ± 0.9	89.3 ± 6.5	86.9 ± 1.0	87.1 ± 4.4	88.5 ± 0.6	88.7 ± 2.5	88.9 ± 1.5	92.6 ± 2.6
LPPGE(T)	**87.9 ± 1.6**	**82.6 ± 6.1**	**85.8 ± 1.2**	**75.8 ± 4.9**	84.0 ± 0.6	84.0 ± 4.3	85.6 ± 1.6	81.0 ± 3.9
LPPGE(A)	84.8 ± 1.2	86.0 ± 6.7	82.1 ± 3.6	80.3 ± 4.5	**83.9 ± 1.4**	**77.1 ± 3.7**	**84.1 ± 1.2**	**77.0 ± 3.5**

Table A4 Result of link prediction for Yale dataset.

(%)

Approach	25%		50%		75%		100%	
	Non-sensitive	Sensitive	Non-sensitive	Sensitive	Non-sensitive	Sensitive	Non-sensitive	Sensitive
GAE	84.6 ± 0.3	89.7 ± 0.7	84.5 ± 0.4	89.1 ± 1.2	84.5 ± 0.5	89.9 ± 1.1	84.4 ± 1.4	90.0 ± 1.2
GAE_RM	83.9 ± 0.3	88.1 ± 0.1	83.9 ± 0.3	86.9 ± 1.0	84.1 ± 0.3	87.7 ± 0.5	84.2 ± 0.2	89.6 ± 0.8
DPNE	39.9 ± 10.4	41.2 ± 12.9	45.7 ± 11.4	47.8 ± 13.5	44.3 ± 12.4	46.0 ± 14.6	48.8 ± 10	52.5 ± 10.2
NPGE	82.0 ± 0.3	85.8 ± 1.0	82.2 ± 0.3	86.4 ± 0.6	82.3 ± 0.4	86.5 ± 0.5	81.7 ± 0.1	85.2 ± 0.6
LPPGE(T)	81.5 ± 0.2	84.5 ± 0.5	80.9 ± 0.4	82.5 ± 0.1	81.4 ± 0.2	82.8 ± 0.6	81.2 ± 0.2	83.1 ± 0.5
LPPGE(A)	**80.1 ± 0.2**	**83.6 ± 0.9**	**80.2 ± 0.2**	**82.4 ± 1.0**	**78.1 ± 0.2**	**78.4 ± 0.9**	**80.2 ± 0.2**	**80.7 ± 1.0**

Table A5 Result of link prediction for Rochester dataset.

(%)

Approach	25%		50%		75%		100%	
	Non-sensitive	Sensitive	Non-sensitive	Sensitive	Non-sensitive	Sensitive	Non-sensitive	Sensitive
GAE	84.9 ± 0.2	89.7 ± 1.0	85.2 ± 0.3	90.3 ± 1.1	85.1 ± 0.3	90.1 ± 1.0	85.1 ± 0.4	90.7 ± 1.1
GAE_RM	84.7 ± 0.5	88.2 ± 1.6	84.8 ± 0.3	89.1 ± 1.2	84.6 ± 0.4	89.2 ± 1.3	84.7 ± 0.3	88.5 ± 0.7
DPNE	54.0 ± 12.7	65.2 ± 18.5	55.8 ± 11.9	67.9 ± 18.0	58.5 ± 8.9	71.2 ± 13.3	60.6 ± 1.2	75.9 ± 3.9
NPGE	83.5 ± 0.3	88.3 ± 1.1	83.4 ± 0.3	88.1 ± 0.7	83.1 ± 0.4	88.7 ± 1.1	83.5 ± 0.4	88.0 ± 1.2
LPPGE(T)	82.1 ± 0.3	81.9 ± 0.9	82.1 ± 0.2	84.4 ± 0.8	81.8 ± 0.4	83.2 ± 0.8	82.2 ± 0.3	83.1 ± 0.7
LPPGE(A)	**79.2 ± 0.1**	**75.4 ± 1.5**	**78.7 ± 0.5**	**76.3 ± 1.4**	**78.4 ± 0.5**	**75.1 ± 1.4**	**80.4 ± 0.3**	**81.3 ± 1.2**

Table A6 Result of node classification for Cora dataset (7 categories).

(%)

Approach	25%		50%		75%		100%	
	MLP	SVM	MLP	SVM	MLP	SVM	MLP	SVM
GAE	73.7	71.5	74.3	71.5	73.1	71.5	74.0	71.5
GAE_RM	72.7	70.1	78.2	73.6	76.3	75.1	75.1	71.3
DPNE	15.2	6.63	14.9	6.63	14.8	6.63	14.5	6.63
NPGE	69.3	60.2	73.0	61.6	73.9	63.9	72.4	68.0
LPPGE(T)	73.6	59.6	75.8	69.8	73.7	54.8	79.2	70.8
LPPGE(A)	72.0	66.9	71.0	61.7	70.7	62.1	73.9	71.8

Table A7 Result of node classification for Citeseer dataset (6 categories).

(%)

Approach	25%		50%		75%		100%	
	MLP	SVM	MLP	SVM	MLP	SVM	MLP	SVM
GAE	59.2	54.1	61.2	54.0	58.9	54.1	58.6	54.0
GAE_RM	58.2	51.9	58.7	54.1	51.9	48.4	63.5	55.5
DPNE	16.7	8.32	17.5	8.32	17.3	8.32	18.6	8.32
NPGE	64.7	56.0	67.3	60.5	67.9	62.6	69.0	64.1
LPPGE(T)	64.1	55.0	66.5	60.7	64.6	57.3	56.4	47.7
LPPGE(A)	60.3	54.7	59.2	57.7	59.6	55.6	66.9	62.3

Table A8 Result of node classification for Yale dataset (6 categories).

(%)

Approach	25%		50%		75%		100%	
	MLP	SVM	MLP	SVM	MLP	SVM	MLP	SVM
GAE	86.5	87.1	86.3	87.1	86.1	87.0	85.8	87.0
GAE_RM	85.2	86.7	85.1	85.8	84.3	86.9	85.1	86.3
DPNE	24.5	4.66	24.6	4.66	24.2	4.66	24.2	4.66
NPGE	83.1	81.4	83.9	82.5	82.8	82.0	78.9	77.6
LPPGE(T)	83.8	82.8	83.8	81.8	84.6	82.8	84.8	83.5
LPPGE(A)	85.4	84.7	83.9	83.0	82.9	81.2	84.0	83.3

Table A9 Result of node classification for Rochester dataset (3 categories).

(%)

Approach	25%		50%		75%		100%	
	MLP	SVM	MLP	SVM	MLP	SVM	MLP	SVM
GAE	84.6	84.1	85.0	84.1	84.6	84.1	84.4	84.1
GAE_RM	85.4	83.4	84.1	83.6	85.1	83.5	85.2	84.2
DPNE	50.4	44.7	50.8	44.7	49.7	44.7	50.2	44.7
NPGE	83.7	81.6	85.0	79.9	85.3	82.6	84.1	80.0
LPPGE(T)	85.4	81.5	85.0	80.9	85.8	82.9	86.2	82.8
LPPGE(A)	83.7	79.7	85.1	79.0	85.1	82.0	86.4	83.6

1009, 2019.

[5] X. Zheng, Z. P. Cai, J. G. Yu, C. K. Wang, and Y. S. Li, Follow but no track: Privacy preserved profile publishing in cyber-physical social systems, *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1868–1878, 2017.

[6] P. Goyal and E. Ferrara, Graph embedding techniques, applications, and performance: A survey, *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[7] Z. B. He, Z. P. Cai, and J. G. Yu, Latent-data privacy preserving with customized data utility for social network data, *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 665–673, 2018.

[8] M. Ellers, M. Cochez, T. Schumacher, M. Strohmaier, and F. Lemmerich, Privacy attacks on network embeddings, arXiv preprint arXiv: 1912.10979, 2019.

[9] P. Drake, Who owns celebrity? Privacy, publicity and the legal regulation of celebrity images, in *Stardom and Celebrity*: *A Reader*, S. Redmond, and S. Holmes, eds. London, UK: SAGE Publications Ltd., 2007, pp. 219–229.

[10] P. Cockerton, David Beckham and Prince William pictured together for campaign against ivory and rhino horn, https://www.mirror.co.uk/news/uk-news/david-beckham-prince-william-pictured-2272167, 2013.

[11] B. Perozzi, R. Al-Rfou, and S. Skiena, DeepWalk: Online learning of social representations, in *Proc. 20th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, New York, NY, USA, 2014, pp. 701–710.

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv: 1301.3781, 2013.

[13] A. Grover and J. Leskovec, node2vec: Scalable feature learning for networks, in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 855–864.

[14] J. Tang, M. Qu, M. Z. Wang, M. Zhang, J. Yan, and Q. Z. Mei, LINE: Large-scale information network embedding, in *Proc. 24th Int. Conf. World Wide Web*, Florence, Italy, 2015, pp. 1067–1077.

[15] D. X. Wang, P. Cui, and W. W. Zhu, Structural deep network embedding, in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 1225–1234.

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

[17] R. van den Berg, T. N. Kipf, and M. Welling, Graph convolutional matrix completion, arXiv preprint arXiv: 1706.02263, 2017.

[18] K. Yang, J. Zhu, and X. Guo, POI neural-rec model via graph embedding representation, *Tsinghua Science and Technology*, vol. 26, no. 2, pp. 208–218, 2021.

[19] A. Korolova, R. Motwani, S. U. Nabar, and Y. Xu, Link privacy in social networks, in *Proc. 17th ACM Conf. Information and Knowledge Management*, Napa Valley, CA, USA, 2008, pp. 289–298.

[20] X. W. Ying and X. T. Wu, On link privacy in randomizing social networks, in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Bangkok, Thailand: Springer, 2009, pp. 28–39.

[21] M. Fire, G. Katz, L. Rokach, and Y. Elovici, Links reconstruction attack, in *Security and Privacy in Social Networks*. New York, NY, USA: Springer, 2013, pp. 181–196.

[22] A. M. Fard and K. Wang, Neighborhood randomization for link privacy in social network analysis, *World Wide Web*, vol. 18, no. 1, pp. 9–32, 2015.

[23] Z. P. Cai, Z. B. He, X. Guan, and Y. S. Li, Collective data-sanitization for preventing sensitive information inference attacks in social networks, *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 577–590, 2018.

[24] D. P. Xu, S. H. Yuan, X. T. Wu, and H. Phan, DPNE: Differentially private network embedding, in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Melbourne, Australia: Springer, 2018, pp. 235–246.

[25] S. Zhang and W. W. Ni, Graph embedding matrix sharing with differential privacy, *IEEE Access*, vol. 7, pp. 89390–89399, 2019.

[26] A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow, Adversarial autoencoders, arXiv preprint arXiv: 1511.05644, 2015.

[27] K. Y. Li, G. C. Luo, Y. Ye, W. Li, S. H. Ji, and Z. P. Cai, Adversarial privacy preserving graph embedding against inference attack, *IEEE Internet of Things Journal*, doi: 10.1109/JIOT.2020.3036583.

[28] A. Pareja, G. Domeniconi, J. Chen, T. F. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson, EvolveGCN: Evolving graph convolutional networks for dynamic graphs, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 4, pp. 5363–5370, 2020.

Kainan Zhang received the BEng degree in computer science from Zhejiang University of Technology, Hangzhou, China in 2013, and the MEng degree in computer engineering from Illinois Institute of Technology, Chicago, IL, USA in 2016. He is currently pursuing the PhD degree at the Department of Computer Science, Georgia State University (GSU), Atlanta, GA, USA. His research interests include privacy preservation, social networking, and machine learning.

Zhi Tian received the BEng degree in electrical engineering from the University of Science and Technology of China, Hefei, China in 1994, the MEng and PhD degrees from George Mason University, Fairfax, VA, USA, in 1998 and 2000, respectively. From 2000 to 2014, she was on the faculty of Michigan Technological University, where she was promoted to a full professor in 2011. From 2011 to 2014, she was a program director at the USA National Science Foundation through an IPA assignment with Michigan Technological University. Since 2015, she has been at the Department of Electrical and Computer Engineering, George Mason University, as a professor. Her general interests lie in the areas of signal processing, wireless communications, and estimation and detection theory. Her current research focuses on compressed sensing for random processes, statistical inference of network data, cognitive radio, and distributed wireless sensor networks. She was an associate editor for the *IEEE Transactions on Wireless Communications* and *IEEE Transactions on Signal Processing*. She is a distinguished lecturer of the IEEE Vehicular Technology Society and the IEEE Communications Society. She received a CAREER Award from the USA National Science Foundation in 2003.

Zhipeng Cai received the BEng degree from Beijing Institute of Technology, Beijing, China in 2001, and the MEng and PhD degrees from University of Alberta, USA in 2004 and 2008, respectively. He is currently an associate professor at the Department of Computer Science, Georgia State University, Atlanta, GA, USA. Prior to joining GSU, he was a research faculty at the School of Electrical and Computer Engineering, Georgia Institute of Technology. His research areas focus on networking and big data. He is the recipient of the NSF CAREER Award.

Daehee Seo received the BEng degree in electrical and electronic engineering from the Dongshin University, Naju, Republic of Korea in 2001, and the MEng and PhD degrees in computing science from the Soonchunhyang University, in 2003 and 2006, respectively. He is currently an assistant professor at Sangmyung University, Seoul, Republic of Korea. His research interests include wireless networks, RegTech, big data security analytics, blockchain security, and cryptography and information theory.