Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 2432351, 15 pages
https://doi.org/10.1155/2022/2432351

Research Article

Graph Embedding-Based Sensitive Link Protection in IoT Systems

Yanfei Lu, Zhilin Deng, Qinghe Gao, and Tao Jing

School of Electronics and Information Engineering, Beijing Jiaotong University, Beijing, China

Correspondence should be addressed to Qinghe Gao; qhgao@bjtu.edu.cn

Received 8 December 2021; Revised 21 February 2022; Accepted 22 March 2022; Published 30 April 2022

Academic Editor: Chunqiang Hu

In the Internet of Things (IoT), massive interconnected intelligent terminal devices constitute diverse networks. Link prediction can serve as a powerful inference attack to speculate the sensitive links in the networks, posing a security threat to entity privacy in IoT. Most antilink prediction methods reduce the prediction ability of link prediction models through link disturbance to hide sensitive links but fail to consider the impact of node attributes on link prediction. This paper proposes a sensitive link protection method based on graph embedding (SLPGE) to combat link prediction attacks. This method is aimed at compressing network topology data into an embedding matrix and lessening private information by combining Variational Graph Autoencoder (VGAE) and Adversarially Regularized Variational Graph Autoencoder (ARVGA). Based on our experiment on two datasets, SLPGE reduces the prediction accuracy of two attack models for sensitive links by up to 30.05% and 15.03% compared to the original data, and the corresponding utility sees a drop of 7.54% and 7.79% at most, which verifies the feasibility of SLPGE—achieving the tradeoff between privacy protection and data utility effectively.

1. Introduction

To build a highly automated, informative, and intelligent system, the Internet of Things (IoT) integrates numerous communication, computing, and sensing devices, ranging from smartphones to vehicles [1], which is an organic collection of intelligent terminal devices and users. In IoT, widely distributed terminal devices establish reliable wireless links through advanced wireless communication and network technology, forming distributed multidomain networks [2]. Networks are ubiquitous in the real world, such as communication networks, social networks, biological networks, and transportation networks, represented by graphs containing nodes and edges. Similarly, the networks in IoT can also be regarded as graphs with terminal devices as nodes and communication links as edges. Although attractive and convenient, IoT also brings a significant challenge, i.e., the concerns on privacy disclosure [3]. As a new paradigm of big data platform, IoT deploys smart city applications to timely monitor, analyze, and respond to volumes of physical data. The data in IoT collected in a distributed manner are strongly correlated with users' sensitive status. However, some information platforms disclose private information inadvertently while trading the data, most likely the graphs in IoT. Furthermore, it does not rule out the possibility that malicious attackers may spy on entity privacy, analyze network traffic, and track users' behavior by stealing the complete network graphs, which invade the entity privacy and threaten the security of the IoT system. At present, the study on privacy for IoT mainly focuses on the privacy of data, identity, and location [4], while rarely mentioning graph privacy, especially the privacy of the communication links between terminal nodes in graphs, i.e., sensitive links. Actually, the disclosure of sensitive links will bring many security threats to the IoT system. For example, some sensitive links usually involve personal privacy, such as the doctor-patient relationship in smart healthcare, one of the typical application scenarios of IoT, and the user trajectories that data requesters may expose when accessing IoT.

In addition, in the man-in-the-middle (MITM) attack, hackers will try to intercept private data; control devices in smart homes, smart industries, and smart healthcare; or destroy the communication links in the IoT system, resulting in privacy disclosure, device failure, and even system collapse, which seriously threaten personal privacy, business activities, and industrial operations. Hence, it is imperative to detach private information from the graphs in advance. The most straightforward operation to hide the sensitive links is to delete the sensitive links in the graphs directly. Unfortunately, sensitive links may be predicted out of released data through data mining techniques, even if they have been deleted [5]. As an essential task in data mining, link prediction has been heating up in recent years. More and more link prediction methods and their application technologies have been proposed. Link prediction can predict the relationship between nodes by mapping the graph information to a continuous vector space. While being widely applied in network analysis, link prediction can also be used as an inference attack to speculate the sensitive links in graphs. Therefore, the data publisher shall carry out privacy processing for the published data to defend link prediction attacks while retaining necessary data utility. In recent years, the privacy disclosure caused by link prediction attacks has attracted researchers' attention, and many researches on antilink prediction have emerged. To defend link prediction based on similarity and deep learning methods, most antilink prediction methods adopt various link disturbances, e.g., random link disturbance, heuristic link disturbance, and evolutionary link disturbance, at the expense of part of data utility [6–13]. Besides, these methods only focus on the graph structure information and fail to consider the unstructured information in graphs, such as node attributes. The node attributes may include the performance, identity, and type of devices, deepening the association strength between nodes and making the attacker's prediction more accurate. As mentioned above, protecting sensitive links against link prediction attacks is an urgent problem to be solved. Significantly, Li et al. [14] proposed an adversarial privacy graph embedding (APGE) method to conceal users' sensitive attributes from inference attacks, which opens up a novel idea for our work. In this paper, we intend to fill this blank by developing a graph embedding-based sensitive link protection method named SLPGE. Our basic idea is to use the graph embedding model combined with Variational Graph Autoencoder (VGAE) and Adversarially Regularized Variational Graph Autoencoder (ARVGA) to encode graph data into an embedding matrix before publishing the data. To be concrete, we utilize adversarial training assisted by two schemes to eliminate private information in the embedding matrix. Then, to balance the tradeoff between privacy and utility, we design the loss functions in SLPGE to retain the utility of graph structure and node labels. The main contributions of this paper are summarized below:

(i) This article focuses on the privacy protection of sensitive links in IoT and proposes a sensitive link protection method (SLPGE) to conceal sensitive links from link prediction attacks

(ii) The results of experiments on two public datasets with node attributes validate that our SLPGE can reduce the prediction accuracy of attack models for sensitive links by 30.05% and 15.03% at most on the basis of the original data

(iii) Our method achieves a tradeoff between privacy and utility. Different from the previous method, our method abandons the idea of directly applying link disturbance on the original graph to remove private information, for which we reduce the loss of utility

The rest of the paper is organized as follows. The related work and preliminaries are reviewed in Sections 2 and 3, respectively. The system models and problem formulation are presented in Section 4. The details of our SLPGE are described in Section 5. The simulation and results are shown in Section 6. Moreover, we give the conclusions and future work in Section 7.

2. Related Work

The emergence of various IoT platforms not only facilitates people's lives but also generates a huge volume of data-carrying personal information. These data can be modeled into graph structure data, and attackers can then easily expose the privacy information hidden in graphs via link prediction. In this section, we briefly introduce the relevant work of graph privacy protection, link prediction, and antilink prediction.

2.1. Graph Privacy Protection. The main methods of graph privacy protection include anonymization, random disturbance, and clustering. Since Sweeney [15] introduced anonymization into graph structure data, different anonymization variants for graphs have also been derived. Ying and Wu [16] disrupted the graph structure by deleting and adding k edges randomly. Li et al. [17] performed spectral clustering according to the distance between nodes firstly and then anonymized subgraphs. For the graphs with node labels, Yuan et al. [18] proposed the protection method of node attribute label K-anonymity to ensure that the labels of at least k nodes are the same. Chester and Srivastava [19] proposed an attribute probability distribution anonymity method to make the probability distribution of the label carried by each node in the attribute sets of its neighbors as close as possible to the global label probability distribution. The random graph modification technology proposed by Hay et al. [20] is the simplest technology to prevent node reidentification and edge exposure. Mittal et al. [9] proposed a link perturbation based on the random walk (LPRW), which improved the privacy and utility of data compared with Hay's method. In edge clustering methods, Liu et al. [21] proposed privacy protection methods for sensitive edge weights in weighted graphs, adopting Gaussian noise disturbance and greedy disturbance. Zheleva and Getoor [22]

mainly considered the privacy of graphs with multiple types of edges and one type of node. Its main idea is to divide the original graph into subgraphs via spectral clustering and then modify the links in the subgraphs and add new links between the subgraphs randomly.

Low data availability and high computational complexity are the common problems of these methods, and their privacy will continue to decrease as inference attacks intensify.

2.2. Link Prediction. Link prediction is aimed at predicting missing facts according to existing entities and has found wide application in social, biological, and communication networks. Known for its powerful inference attack, link prediction has been maliciously used to spy on the privacy of entities in the networks. Among plenty of link prediction methods, classification models such as support vector machine (SVM) [23], multilayer perceptron (MLP) [24], and k nearest neighbor (KNN) [25] regard link prediction as a binary classification problem, in which the connected node pairs and unconnected node pairs are regarded as positive samples and negative samples, respectively.

2.3. Antilink Prediction. At present, most antilink prediction methods for graph structure data disturb the graph structure by adding some new links and deleting part of nonsensitive links strategically to reduce the prediction ability of various link prediction methods and achieve the privacy protection of sensitive links. Liu and Terzi [6] proposed to achieve k-degree anonymization through edge addition or deletion strategies. Rousseau et al. [7] proposed two approaches that preserve the coreness of a graph while anonymizing it through various edge modification operations. Fard and Wang [8] and Mittal et al. [9] proposed two structure-aware randomization perturbation methods based on local perturbation and random walk considering the structural proximity of nodes. Zhou et al. [10] regarded the links between the end nodes of a sensitive link and their common nodes as the candidate links to be deleted and expressed the attack on local similarity as an optimization problem to determine which links to delete. Chen et al. [11] proposed an iterative gradient attack (IGA) method based on integral gradient information in Graph Autoencoder (GAE). The gradients obtained by maximizing the loss of sensitive links represent the influence of other links on sensitive links. During k iterations, n links with the largest gradients are modified. Yu et al. [12] combated resource allocation (RA) indicator link prediction via random, heuristic, and evolutionary link disturbance. Among these three methods, random link disturbance increases and changes links without any strategy, heuristic link disturbance reduces the link prediction ranking of node pairs in the test set, and evolutionary link disturbance selects the links to be added and deleted according to the fitness function. Waniek et al. [13] selected to delete or add the most influential links to hide sensitive links by reducing or creating the closed triangles containing sensitive links.

The methods mentioned above can be used in IoT systems to avoid the leakage of sensitive links in data transactions. However, two shortcomings are present in the above methods: the first is that the utility of the graph will be lost due to link disturbance, and the second is that they lack the consideration of the impact of node attributes on link prediction.

3. Preliminaries

As a kind of non-Euclidean data, a graph is difficult to be directly processed by traditional data analysis methods or deep learning models such as Convolutional Neural Network (CNN) [26] and Recurrent Neural Network (RNN) [27] due to the high computational and space overhead. Graph embedding, also called network representation learning, is aimed at mapping graph data, usually a high-dimensional dense matrix to low-dimensional dense vectors. Graph embedding has more flexible and rich calculation methods to apply deep learning models directly for graph analysis tasks. Graph Neural Network (GNN) represents the deep learning method of graph embedding. By modeling the nodes and communication links in the networks, GNN can be applied to solve the privacy disclosure problem in IoT. For the advantages of feature extraction from non-Euclidean data, our SLPGE is based on some GNN models. In this section, the GNN models involved in SLPGE, e.g., Graph Convolutional Network (GCN), VGAE, and ARVGA, are briefly introduced. For the sake of clarity, the frequently used notations and their meanings are listed in Table 1.

3.1. Graph Convolutional Network. In 2013, Bruna et al. [28] first proposed the neural network on the graph and gave two structures based upon a hierarchical clustering of the domain and the spectrum of the graph Laplacian. As a typical GNN model, GCN [29] is a scalable approach for semisupervised learning on graph data, which uses the spectrum of the graph Laplacian to achieve convolution on graphs. After each convolution of GCN, the node features are the weighted sum of the previous features of the nodes and their neighbor nodes, for which the nodes can aggregate further features with the deepening of layers. Hence, the superiority of GCN is to incorporate local graph structure and node features naturally. Suppose the adjacency matrix $A \in \mathbb{R}^{N\times N}$ represents the connection relationship between n nodes, then the layer-wise propagation rule of GCN is as follows:

$$H_{l+1} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H_l W_l\right), \tag{1}$$

where $\mathbf{H}_l$ is the feature matrix of the l^{th} layer, $\mathbf{W}_l$ is the trainable weight matrix, and $\sigma(\cdot)$ is an activation function. $\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ is the normalization of $\tilde{A}$ where $\tilde{A} = \mathbf{A} + \mathbf{I}_N$, $\mathbf{I}_N \in \mathbb{R}^{N\times N}$ is the identity matrix, $\tilde{D}$ is the degree matrix of $\tilde{A}$, and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. The degree of a node is the number of first-order neighbors connected to the node. Equation (1) can be abbreviated as $\mathbf{H}_{l+1} = f(\mathbf{H}_l, \mathbf{A})$, for $\mathbf{A}$ is the input of each layer.

Table 1: Summary of notations.

Notations	Meanings
$\mathbf{G}$	The undirected original graph
$\mathbf{V}$	The set of nodes in $\mathbf{G}$
$\lvert\mathbf{V}\rvert$	The number of nodes
$\mathbf{E}$	The set of edges in $\mathbf{G}$
$\lvert\mathbf{E}\rvert$	The number of edges
v_i	The i^{th} node
e_{ij}	The edge between v_i and v_j
$\mathbf{X}$	The node feature matrix of $\mathbf{V}$
F	The number of node attributes
$\mathbf{A}$	The adjacency matrix of $\mathbf{G}$
$\mathbf{A}_p$	The adjacency matrix of privacy graph
$\mathbf{A}_t$	The adjacency matrix of training graph
$\widehat{\mathbf{A}}$	The reconstructed adjacency matrix of $\mathbf{G}$
$\widehat{\mathbf{A}}_p$	The reconstructed adjacency matrix of privacy graph
A_{ij}	The link state between v_i and v_j in $\mathbf{A}$
$\widehat{A}_{ij}$	The link state between v_i and v_j in $\widehat{\mathbf{A}}$
L	The number of categories for node labels
$\widehat{\mathbf{y}}$	The node label matrix predicted by softmax classifier with each row includes the predicted values of L categories
$\mathbf{Z}_p$	The privacy embedding of privacy graph
$\mathbf{Z}_f$	The link protection graph embedding
$\mathbf{Z}$	The higher dimensional graph embedding concatenated by $\mathbf{Z}_f$ and $\mathbf{Z}_p$
m	The maximum number of edges added for each sensitive link
$\mathbf{E}_{\text{sl}}$	The sensitive links in $\mathbf{G}$
$\mathbf{E}_{\text{nsl}}$	Part of nonsensitive links in $\mathbf{G}$
$\mathbf{E}_{\text{know}}$	The links which are known to the attack models
L_{link}	The reconstruction loss
L_{lable}	The node classification loss
L_{g}	The distribution loss of the generator
L_{G}	The total loss of the generator
L_{D}	The distribution loss of the discriminator
Acc_{sl}	The classification accuracy of the attack models for sensitive links
Acc_{nsl}	The classification accuracy of the attack models for nonsensitive links
$\text{Acc}_{\text{recon}}$	The link reconstruction accuracy of $\mathbf{Z}_f$
$\text{Re c}_{\text{recon}}$	The link reconstruction recall of $\mathbf{Z}_f$
Acc_{node}	The node classification accuracy of $\mathbf{Z}_f$

3.2. Variational Graph Autoencoders. Soon after the proposal of GCN, to expand the capability of GCN, VGAE proposed by Kipf and Welling [30] adopts GCN as an encoder to generate specific graph embedding for different tasks of the graph, not limited to node classification. VGAE is an unsupervised learning framework derived from Variational Autoencoders (VAE) [31], which obtains graph embedding through the encoder-decoder structure. VGAE consists of a two-layer GCN encoder and a simple inner-product decoder. The two-layer GCN can be defined as follows:

$$\begin{aligned}\text{GCN}(\mathbf{X}, \mathbf{A}) &= f(\mathbf{H}_1, \mathbf{A}) = \sigma\left(\bar{\mathbf{A}} f(\mathbf{H}_0, \mathbf{A})\mathbf{W}_1\right) \\ &= \bar{\mathbf{A}}\text{ReLU}\left(\bar{\mathbf{A}}\mathbf{H}_0\mathbf{W}_0\right)\mathbf{W}_1,\end{aligned} \tag{2}$$

where $\bar{\mathbf{A}} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ is the symmetrically normalized adjacency matrix and $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$ is the activation function of the first layer. $\sigma(\cdot)$ of the second layer is determined according to the specific task. The encoder of VGAE is aimed at learning the mean μ and the standard deviation σ of a multidimensional Gaussian distribution from which the graph embedding $\mathbf{Z}$ is sampled. The process is briefly described below:

$$\begin{aligned} \mu &= \mathrm{GCN}_{\mu}(\mathbf{X}, \mathbf{A}), \\ \log \sigma &= \mathrm{GCN}_{\sigma}(\mathbf{X}, \mathbf{A}), \\ \mathbf{Z} &= \mu + \varepsilon \times \sigma, \end{aligned} \tag{3}$$

where $\mathbf{X} \in \mathbb{R}^{N \times F}$ replaces $\mathbf{H}_0$ in Equation (2) as the node feature matrix of the first layer and $\mathrm{GCN}_{\mu}(\mathbf{X}, \mathbf{A})$ and $\mathrm{GCN}_{\sigma}(\mathbf{X}, \mathbf{A})$ share first-layer parameters $\mathbf{W}_0$. $\mathbf{Z} \sim \mathcal{N}(\mu, \sigma^2)$ is the graph embedding matrix and $\varepsilon \sim \mathcal{N}(0, 1)$ is the noise sampled from the standard Gaussian distribution. The inner product is used as a decoder in VGAE, and the formula is as follows:

$$\widehat{\mathbf{A}} = \sigma\left(\mathbf{Z} \cdot \mathbf{Z}^T\right), \tag{4}$$

where $\sigma(\cdot) = 1/1 + \exp(-\cdot)$ is the sigmoid function. $\widehat{\mathbf{A}}$ is the reconstructed adjacency matrix, and $\widehat{A}_{ij}$ can be regarded as the product of independent event probabilities of the i^{th} node and the j^{th} node. When $\widehat{A}_{ij}$ is greater than the threshold 0.5, it means that there is a link between the i^{th} node and the j^{th} node.

VGAE has two optimization objectives: one is to make $\widehat{A}$ and A as similar as possible; the other is to make the distribution of Z as close to the standard Gaussian distribution as possible. Since binary cross-entropy (BCE) can determine the proximity between the actual output and the expected output and Kullback-Leibler (KL) divergence can measure the difference between two distributions, the loss function of VGAE composed of BCE and KL divergence can be expressed as

$$\mathrm{loss} = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}[\log p(\mathbf{A} \mid \mathbf{Z})] - \mathrm{KL}[q(\mathbf{Z} \mid \mathbf{X}, \mathbf{A}) \| p(\mathbf{Z})]. \tag{5}$$

Here, the former minimizes the reconstruction loss through the cross-entropy function, and the latter minimizes the KL divergence. $p(\mathbf{A} \mid \mathbf{Z}) = \sigma(\mathbf{Z} \cdot \mathbf{Z}^T)$, $q(\mathbf{Z} \mid \mathbf{X}, \mathbf{A}) = \prod_{i=1}^{N} q(\mathbf{z_i} \mid \mathbf{X}, \mathbf{A}) = \prod_{i=1}^{N} \mathcal{N}(\mathbf{z_i} \mid \mu_{\mathbf{i}}, \mathrm{diag}(\sigma_{\mathbf{i}}^2))$ is the real distribution function we get, and $p(\mathbf{Z}) = \prod_i p(\mathbf{z_i}) = \prod_i \mathcal{N}(\mathbf{z_i} \mid 0, \mathbf{I})$ is a Gaussian prior. $\mathrm{KL}[q(\cdot) \| p(\cdot)]$ is the KL divergence between $q(\cdot)$ and $p(\cdot)$. We expect $q(\mathbf{Z} \mid \mathbf{X}, \mathbf{A})$ to be as close to $p(\mathbf{Z})$ as possible.

More specifically, $\mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}[\log p(\mathbf{A} \mid \mathbf{Z})]$ in Equation (5) can be abbreviated as $\mathrm{loss}_{\mathrm{link}}$ below:

$$\mathrm{loss}_{\mathrm{link}} = -\frac{1}{|\mathbf{V}|^2} \sum_{i \in \mathbf{V}} \sum_{j \in \mathbf{V}} \left(p_1 A_{ij} \log \widehat{A}_{ij} + \left(1 - A_{ij}\right) \log\left(1 - \widehat{A}_{ij}\right)\right), \tag{6}$$

where A_{ij} represents the value which is 0 or 1 of an element in A, $\widehat{A}_{ij}$ represents the probability value of the corresponding element in $\widehat{A}$, and p_1 is the ratio of the number of 0 to 1 in A, which can be used to solve the problem of imbalance between positive and negative samples. $\mathrm{KL}[q(\mathbf{Z} \mid \mathbf{X}, \mathbf{A}) \| p(\mathbf{Z})]$ in Equation (5) can be abbreviated as $\mathrm{loss}_{\mathrm{dist}}$ below:

$$\mathrm{loss}_{\mathrm{dist}} = -\frac{1}{2}\left(1 + \log \sigma^2 - \mu^2 - \sigma^2\right). \tag{7}$$

3.3. Adversarially Regularized Variational Graph Autoencoder. To force the graph embedding learned by VGAE to fit the prior distribution better, Pan et al. [32] proposed ARVGA by combining VGAE and Generative Adversarial Network (GAN). GAN was first proposed by Goodfellow et al. [33] to serve as a generative model bridging supervised learning and unsupervised learning in 2014. Most recently, exploiting GAN to work out elegant solutions to severe privacy and security problems has become increasingly popular in both academia and industry due to its game theoretic optimization strategy [34]. Typically, GAN consists of a generator G and a discriminator D, the purpose of which is to mix the spurious with the genuine in a nutshell. During the iterative training, G is trained to generate the fake samples to convince D that the fake samples come from a prior data distribution, while D discriminates whether an input sample comes from the prior data distribution or G we built. In ARVGA, we take VGAE as G, a two-layer fully connected network as D where the output layer only has one dimension with a sigmoid function. The equation for training the encoder model with the discriminator can be written as follows:

$$\min_G \max_D \left(\mathbb{E}_{x \sim p_{\mathrm{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]\right). \tag{8}$$

Here, $x \sim P_{\mathrm{data}}(x)$ is the real sample, $z \sim P_z(z)$ is the original data, $G(z)$ is the fake sample, and $D(\cdot)$ is the probability that the sample is true. G is aimed at minimizing the equation while D is aimed at the opposite of G. Through the game between G and D, ARVGA can enforce the graph embedding to match the prior distribution and produce a robust representation.

4. Model and Problem Formulation

In this article, our work is based on the following assumptions in the graph of IoT: The connections between devices are bidirectional. There are L types of devices in the graph, and each device has its own attribute information such as internal storage, bandwidth, and hard disk. Sensitive links are the links that need to be hidden, while nonsensitive links are those which can be made public. The links whose end nodes have a larger total degree are defined as sensitive links. The nodes with larger degrees usually have more influence in the graph, so the links between these nodes are also more meaningful. Moreover, we take SVM and MLP as attack models to test the performance of our method, and part of nonsensitive links and nonexistent links in the graph are known to the attack models.

4.1. Network Model. We express one of the graphs of IoT as an undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X})$. $\mathbf{V} = \{v_1, v_2, \cdots, v_N\}$ is the set of n terminal nodes and $N = |\mathbf{V}|$. $\mathbf{E}$ contains the edges e_{ij} with the communication link between v_i and $v_j (1 \le i, j \le N)$, including sensitive links and nonsensitive links. $\bar{\mathbf{E}}$ is the set of nonexistent links and $\mathbf{E} \cup \bar{\mathbf{E}} = \mathbf{E}_{N^2}$, where $\mathbf{E}_{N^2}$ contains $n \times n$ edges that can be connected by n nodes. Node attributes are summarized in a feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$ with the i^{th} row representing the attributes of v_i and F is the number of attributes. $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix, where $A_{ij} = 1$ if $e_{ij} \in \mathbf{E}$; otherwise, $A_{ij} = 0$. $\mathbf{E}_{\text{sl}} \subset \mathbf{E}$ is the set of sensitive links, $\mathbf{E}_{\text{nsl}} \subset \mathbf{E}$ is the set of nonsensitive links, and $E_{\text{sl}} \cap E_{\text{nsl}} = \varnothing$.

4.2. Attack Model. Both SVM and MLP have strong classification abilities for nonlinear problems with different structures.

SVM is a classification model based on the structural risk minimization criterion in machine learning. For the nonlinear classification problems, SVM adopts a nonlinear function $\phi(x)$ to map the samples from the input space to a high-dimensional feature space where the samples are linearly separable and construct an optimal classification hyperplane to categorize new samples utilizing labeled training data. Given the training set $T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_k, y_k)\} (x_i \in \mathbb{R}^N)$, SVM can transform the classification problem into a convex quadratic optimization problem as follows:

$$\begin{cases} \min\limits_{\alpha} \dfrac{1}{2} \sum\limits_{i=1}^{k} \sum\limits_{j=1}^{k} \alpha_i \alpha_j y_i y_j \phi(x_i) \phi(x_j) - \sum\limits_{i=1}^{k} \alpha_i \\ \text{s.t.} \sum\limits_{i=1}^{k} \alpha_i y_i = 0 \\ 0 \le \alpha_i \le C, i = 1, 2, \cdots, k, \end{cases} \tag{9}$$

where α_i is a Lagrange multiplier and C is the penalty factor. Since the computation of $\phi(x_i) \times \phi(x_j)$ increases sharply in the high-dimensional space, SVM introduces kernel function $K(x_i, x) = \phi(x_i) \cdot \phi(x)$ to avoid the problem. The kernel function we choose is Gaussian kernel:

$$K(x_i, x) = \exp\left(-\frac{\|x_i - x\|^2}{2\sigma^2}\right), \tag{10}$$

where σ^2 is the variance. In this case, the classification decision function is as follows:

$$f(x) = \text{sign}\left(\sum_{i=1}^{k} \alpha_i y_i \exp\left(-\frac{\|x_i - x\|^2}{2\sigma^2}\right) + b\right), \tag{11}$$

where b is the bias constant.

MLP is a fully connected artificial neural network, consisting of an input layer, hidden layer, and output layer. MLP adjusts the parameters in the hidden layer units through the supervised back propagation (BP) algorithm and gradient descent algorithm to reduce the error between the actual output and the expected output. The forward propagation mechanism of MLP is expressed as below:

$$\mathbf{H}^{(l+1)} = \sigma\left(\mathbf{W}^{(l)} \mathbf{H}^{(l)} + \mathbf{b}^{(l)}\right), \tag{12}$$

where $\mathbf{H}^{(l)}$ is the input matrix, $\mathbf{W}^{(l)}$ is the weight matrix, $\mathbf{b}^{(l)}$ is the bias, and $\mathbf{H}^{(l+1)}$ is the output of the hidden layer. Thus, the decision function of MLP with only one hidden layer can be expressed as follows:

$$f(x) = \sigma\left(\mathbf{W}^{(1)}\left(\sigma\left(\mathbf{W}^{(0)}\mathbf{x} + \mathbf{b}^{(0)}\right)\right) + \mathbf{b}^{(1)}\right), \tag{13}$$

where $\mathbf{x}$ is the input and $\sigma(\cdot)$ is an activation function.

4.3. Problem Formulation. Given a graph $\mathbf{G}$, our model will compress it into a graph embedding $\mathbf{Emb}$ where the i^{th} row represents the vector emb_i of v_i. The vector of e_{ij} can be expressed as $\text{emb}_{ij} = (\text{emb}_i, \text{emb}_j)$. Suppose a link set $\mathbf{E}_{\text{know}}$ containing k nonsensitive links (class 1) and k nonexistent links (class 0) in $\mathbf{G}$ have been exposed to attackers. Then, the embedding matrix of $\mathbf{E}_{\text{sl}}$ and $\mathbf{E}_{\text{know}}$ are $\mathbf{Emb}_{\mathbf{E}_{\text{sl}}}$ and $\mathbf{Emb}_{\mathbf{E}_{\text{know}}}$ where each row represents an edge embedding vector.

During data transactions, attackers will collect or steal $\mathbf{Emb}$ by any means to infer sensitive links through link prediction. Our goal is to achieve the balance between privacy protection and data utility. To this end, we use "minmax" strategy to maximize the distance between the predicted label $\text{label}_{\text{pred}}$ of sensitive links and its real label $\text{label}_{\text{real}}$ and then minimize the distance between $\text{label}_{\text{pred}}$ of nonsensitive links and its $\text{label}_{\text{real}}$. The mathematical description is as follows:

$$\begin{aligned} &\text{Training} : \text{Clf\_fit}\left(\mathbf{Emb}_{\mathbf{E}_{\text{know}}}, \text{label}_{\mathbf{E}_{\text{know}}}\right), \\ &\text{Prediction} : \text{label}_{\text{pred}} = \text{Clf}_{\text{predict}}\left(\mathbf{Emb}_{\mathbf{E}_{\text{sl}}}\right), \\ &\text{Objective} : \min_{E_{\text{nsl}}} \max_{E_{\text{sl}}} \left|\left|\text{label}_{\text{pred}} - \text{label}_{\text{real}}\right|\right|^2, \end{aligned} \tag{14}$$

where $\text{Clf\_fit}(x_{\text{train}}, y_{\text{train}})$ means to fit the classifier model with the training data and $\text{Clf\_predict}(x_{\text{test}})$ means to predict the labels of x_{test}. $\text{label}_{\mathbf{E}_{\text{know}}}$ is the label set of $\mathbf{E}_{\text{know}}$ where k ones represent nonsensitive links and k zeros represent nonexistent links. We expect to get a graph embedding which can work for our purpose.

5. Algorithm

The SLPGE framework consists of two parts. In this section, we will introduce the framework of SLPGE in Subsections 5.1 and 5.2, and the evaluation indicators are described in Subsection 5.3.

5.1. Generate the Privacy Embedding $\mathbf{Z}_p$. Part 1 is to generate a privacy embedding $\mathbf{Z}_p$. In order to put more privacy

Input: $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X})$: the original graph
$\mathbf{A}$: the adjacency matrix of $\mathbf{G}$
$\mathbf{E}_{sl}$: the set of sensitive links
$\mathbf{N}_i$: the neighbor nodes set of v_i
m: the maximum number of edges added for each sensitive link
Output: $\mathbf{A}_p$: the adjacency matrix of privacy graph
1:**for**$sl_{ij} \in \mathbf{E}_{sl}$**do**
2: find the neighbor nodes sets $\mathbf{N}_i$ and $\mathbf{N}_j$
3: **for**$n_i \in \mathbf{N}_i, n_j \in \mathbf{N}_j$**do**
4: **if**$A_{n_i n_j} = 1$ and $A_{n_i v_j} = 0 (A_{n_j v_i} = 0)$**then**
5: **if** the number of edges added for $sl_{ij} \le m$**then**
6: $A_{n_i v_j} = 1 (A_{n_j v_i} = 1)$
7: **end if**
8: **end if**
9: **end for**
10:**end for**
11:**return** take the modified $\mathbf{A}$ as $\mathbf{A}_p$

Algorithm 1: Generate privacy graph by adding edges.

information into $\mathbf{Z}_p$, we first change the structure of the original graph $\mathbf{G}$ to enhance the connection strength of end nodes of sensitive links. Before inputting the graph data into the model, we preprocess $\mathbf{G}$ through Algorithms 1 and 2 corresponding to Figures 1 and 2. For Algorithm 1, we believe that two nodes with more common neighbors have a closer relationship. As shown in Figure 1, e_{01} is a sensitive link, $\{v_2, v_4\}$ and $\{v_3, v_5\}$ are the neighbor sets of v_0 and v_1, respectively, and e_{23} exists; in this case, we link e_{03} and e_{12} to make the relationship between v_0 and v_1 closer. For Algorithm 2, we believe that the main information in the graph will focus on sensitive links when other irrelevant nodes and links are removed. As shown in Figure 2, we only keep the sensitive links and their adjacent links to retain the information about the sensitive links to the greatest extent. The two privacy graph adjacency matrices $\mathbf{A}_p$'s obtained by Algorithms 1 and 2 are, respectively, used as the input of the encoder to output two $\mathbf{Z}_p$'s.

For VGAE is more robust and suitable for small graphs, we adopt VGAE to obtain $\mathbf{Z}_p$ in this part as shown in Figure 3. As discussed in Section 3, the mechanism for the encoder to generate $\mathbf{Z}_p$ can refer to Equations (2) and (3). Then, we get the reconstructed adjacency matrix $\widehat{\mathbf{A}}_p = \text{sigmoid}(\mathbf{Z}_p \cdot \mathbf{Z}_p{}^T)$. The reconstruction loss L_{link} is the same as Equation (5), except that $\mathbf{A}$ and $\widehat{\mathbf{A}}$ are replaced by $\mathbf{A}_p$ and $\widehat{\mathbf{A}}_p$.

For node classification, a softmax classifier is followed by the encoder to predict the labels of the nodes. The node classification loss function L_{label} is as follows:

$$L_{\text{label}} = -\sum_{i=1}^{|\mathbf{V}|} \sum_{l=1}^{L} y_{il} \ln \widehat{y}_{il}, \tag{15}$$

where y_{il} represents the real label of v_i in category l with a value of 0 or 1, while $\widehat{y}_{il}$ is the value we predict in $\widehat{\mathbf{y}}$ and $\widehat{y}_{il}$

Input: $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X})$: the original graph
$\mathbf{A}$: the adjacency matrix of $\mathbf{G}$
$\mathbf{E}_{sl}$: the set of sensitive links
$\mathbf{V}_{sl}$: the end nodes set of $\mathbf{E}_{sl}$
$\mathbf{A}_p$: the empty matrix with the same shape as $\mathbf{A}$
$\mathbf{N}$: the neighbor nodes set
Output: $\mathbf{A}_p$: the adjacency matrix of privacy graph
for$v \in \mathbf{V}_{sl}$**do**
2: find the neighbor nodes set $\mathbf{N}$ of v
for$n \in \mathbf{N}$**do**
4: $\mathbf{A}_p vn = 1 (\mathbf{A}_p nv = 1)$
end for
6: end for
return$\mathbf{A}_p$

Algorithm 2: Generate privacy graph by deleting edges.

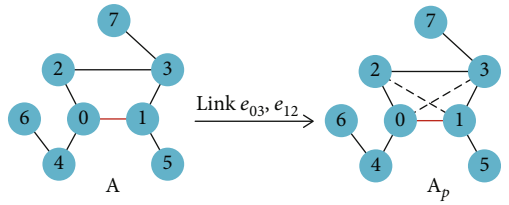

Figure 1: Generate the privacy graph with edge added.

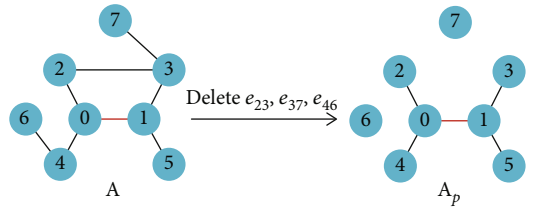

Figure 2: Generate the privacy graph with edge deleted.

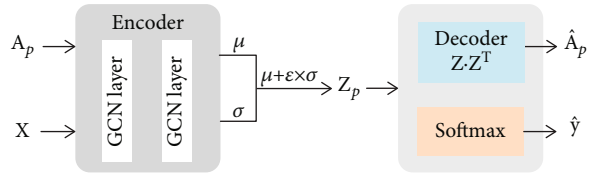

Figure 3: Generate the privacy graph embedding $\mathbf{Z}_p$.

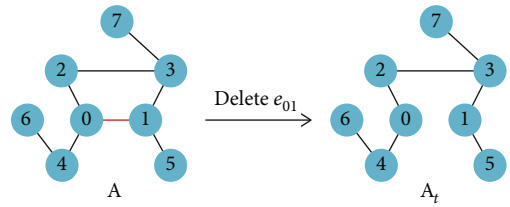

Figure 4: Generate the training graph.

$= \text{softmax}((\mathbf{Z}_p)_{il}) = 1/\mathcal{Z} \exp((\mathbf{Z}_p)_{il})$ with $\mathcal{Z} = \sum_{l=1}^{L} e^{(\mathbf{Z}_p)_{il}}$. Therefore, the total loss of Part 1 is as follows:

$$L_1 = L_{\text{link}} + L_{\text{label}}. \tag{16}$$

Through the BP mechanism of L_1, we train the encoder to generate $\mathbf{Z}_p$ that contains privacy information and conforms to a prior distribution.

5.2. Generate the Link Protection Graph Embedding $\mathbf{Z}_f$. Part 2 generates a graph embedding $\mathbf{Z}_f$ that can protect sensitive links. In order to reduce the most intuitive privacy information, we remove the sensitive links in $\mathbf{A}$ to obtain $\mathbf{A}_t$, the adjacency matrix of the training graph, as shown in Figure 4. The model in this part is designed based on ARVGA, as shown in Figure 5. The inputs of the encoder are $\mathbf{A}_t$ and $\mathbf{X}$. $\mathbf{Z}_f$ output from the encoder is the input of the discriminator and the softmax classifier. Unlike Part 1, $\mathbf{Z}_f$ and $\mathbf{Z}_p$ are combined by adding or concatenating to form a higher dimensional matrix $\mathbf{Z}$ as the input of the inner-product decoder. The node classification loss function L_{label} is the same as Equation (15). In order to distinguish the two $\mathbf{Z}_p$'s obtained in Part 1, in Section 6.2, we will use **SLP GE**$^+$ to explain that Algorithm 1 is used for the generation of $\mathbf{Z}_p$ and **SLPGE**$^-$ to explain that Algorithm 2 is used.

Since the adversarial training between the encoder and the discriminator can force $\mathbf{Z}_f$ to match a prior distribution, the KL divergence in Equation (5) is omitted. Here, we try to use Mean Squared Error (MSE) as the reconstruction loss L_{link}, and the reconstruction target is changed to $\mathbf{A}$:

$$L_{\text{link}} = -\frac{1}{|\mathbf{V}|} \sum_{i \in \mathbf{V}} \sum_{j \in \mathbf{V}} \left\| A_{ij} - \hat{A}_{ij} \right\|^2. \tag{17}$$

In the discriminator, we take $\mathbf{Z}_f$ as the fake samples and Gaussian distribution samples as the real samples, then input them into the discriminator, i.e., a two-layer full connection layer network to get two estimated value d_{fake} and d_{real}, respectively. L_{g} and L_{D} are the distribution loss of the generator and the discriminator, which are both calculated by BCE:

$$L_{\text{g}} = -\log(d_{\text{fake}}), \tag{18}$$

$$L_{\text{D}} = -\log(d_{\text{real}}) - \log(1 - d_{\text{fake}}). \tag{19}$$

Therefore, the total loss of the generator can be written as follows:

$$L_{\text{G}} = L_{\text{link}} + L_{\text{label}} + L_{\text{g}}. \tag{20}$$

Through the adversarial training, the discriminator learns how to distinguish between the real samples and the fake samples, while the generator learns to generate a better $\mathbf{Z}_f$ to confuse the discriminator. In general, the training process of obtaining $\mathbf{Z}_f$ can be summarized as Algorithm 3.

5.3. Evaluation Indicators. This subsection will introduce the quantitative indicators of privacy and utility.

5.3.1. Privacy. Our chief target is to reduce the prediction accuracy of the attack models for sensitive links. Input an embedding vector of a sensitive or nonsensitive link to the attack models; if the predicted value is 1, it means the link exists and vice versa. Privacy is measured by the prediction accuracy Acc_{sl} of the attack models for sensitive links:

$$\text{Acc}_{\text{sl}} = \frac{N_{\text{sl}}}{|\mathbf{E}_{\text{sl}}|} \times 100\%, \tag{21}$$

where N_{sl} is the number of sensitive links predicted to exist and $|\mathbf{E}_{\text{sl}}|$ is the total number of sensitive links. When the security of $\mathbf{Z}_f$ is stronger, Acc_{sl} is lower.

5.3.2. Utility. Utility includes three parts: the prediction accuracy of the attack models for nonsensitive links, the accuracy and recall of the reconstructed graph, and the accuracy of node classification. Taking the existing links as positive samples and the nonexistent links as negative samples, the quantitative expression of utility is as follows:

$$\text{Acc}_{\text{nsl}} = \frac{N_{\text{nsl}}}{|\mathbf{E}_{\text{nsl}}|} \times 100\%, \tag{22}$$

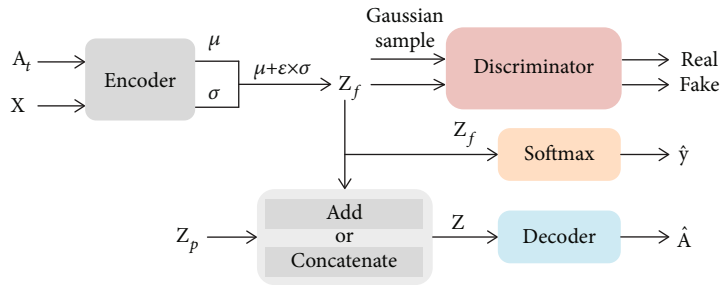

Figure 5: Generate the link protection graph embedding $\mathbf{Z}_f$.

Input: $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X})$: the original graph
$\mathbf{A}_t$: the adjacency matrix of training graph
$\mathbf{Z}_p$: the privacy embedding
Output: $\mathbf{Z}_f$: the link protection graph embedding
for each epoch **do**
Generate the adjacency matrix $\mathbf{A}_t$ of training graph
Input $\mathbf{A}_t$ and $\mathbf{X}$ to the encoder to generate $\mathbf{Z}_f$
Input $\mathbf{Z}_f$ to the softmax classifier
Adding or concatenating $\mathbf{Z}_f$ and $\mathbf{Z}_p$ to form $\mathbf{Z}$
Input $\mathbf{Z}$ to the inner-product decoder
Input $\mathbf{Z}_f$ and the Gaussian samples to the discriminator
Update the generator by minimizing L_G
Update the discriminator by minimizing L_D
end for
return $\mathbf{Z}_f$

Algorithm 3: Generate link protection graph embedding.

where $\mathrm{Acc}_{\mathrm{nsl}}$ is the prediction accuracy of nonsensitive links, N_{nsl} is the number of nonsensitive links predicted to exist, and $|\mathbf{E}_{\mathrm{nsl}}|$ is the total number of nonsensitive links:

$$\mathrm{Acc}_{\mathrm{recon}} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \times 100\%, \tag{23}$$

$$\mathrm{Re}\ c_{\mathrm{recon}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \times 100\%, \tag{24}$$

where $\mathrm{Acc}_{\mathrm{recon}}$ is the ratio of existing links and nonexistent links that are reconstructed correctly and $\mathrm{Re}\ c_{\mathrm{recon}}$ represents how many existing links have been reconstructed. TP and FP are the numbers of reconstructed positive and negative samples, and FN and TN are the numbers of nonreconstructed positive and negative samples:

$$\mathrm{Acc}_{\mathrm{node}} = \frac{N_{\mathrm{node}}}{|\mathbf{V}|} \times 100\%, \tag{25}$$

where $\mathrm{Acc}_{\mathrm{node}}$ is the ratio of the nodes classified correctly to the total number of nodes. N_{node} is the number of nodes classified correctly, and $|\mathbf{V}|$ is the number of nodes. Our tradeoff is protecting privacy while preserving utility, that is, reducing $\mathrm{Acc}_{\mathrm{sl}}$ and keeping $\mathrm{Acc}_{\mathrm{nsl}}$, $\mathrm{Acc}_{\mathrm{recon}}$, $\mathrm{Re}\ c_{\mathrm{recon}}$, and $\mathrm{Acc}_{\mathrm{node}}$ high.

Table 2: Details of experiment.

Parameters	Cora	Yale
$\lvert\mathbf{V}\rvert$	2708	5278
$\lvert\mathbf{E}\rvert$	5278	405450
F	1433	188
L	7	7
$\lvert\mathbf{E}_{\mathrm{sl}}\rvert$	100	200
$\lvert\mathbf{E}_{\mathrm{nsl}}\rvert$	100	200
$\lvert\mathbf{E}_{\mathrm{know}}\rvert$	400	400
m	10	15
$\mathbf{Z}_p$	8-dim	7-dim
$\mathbf{Z}_f$	8-dim	7-dim
$\mathbf{Z}$(add)	8-dim	7-dim
$\mathbf{Z}$(cat)	16-dim	14-dim

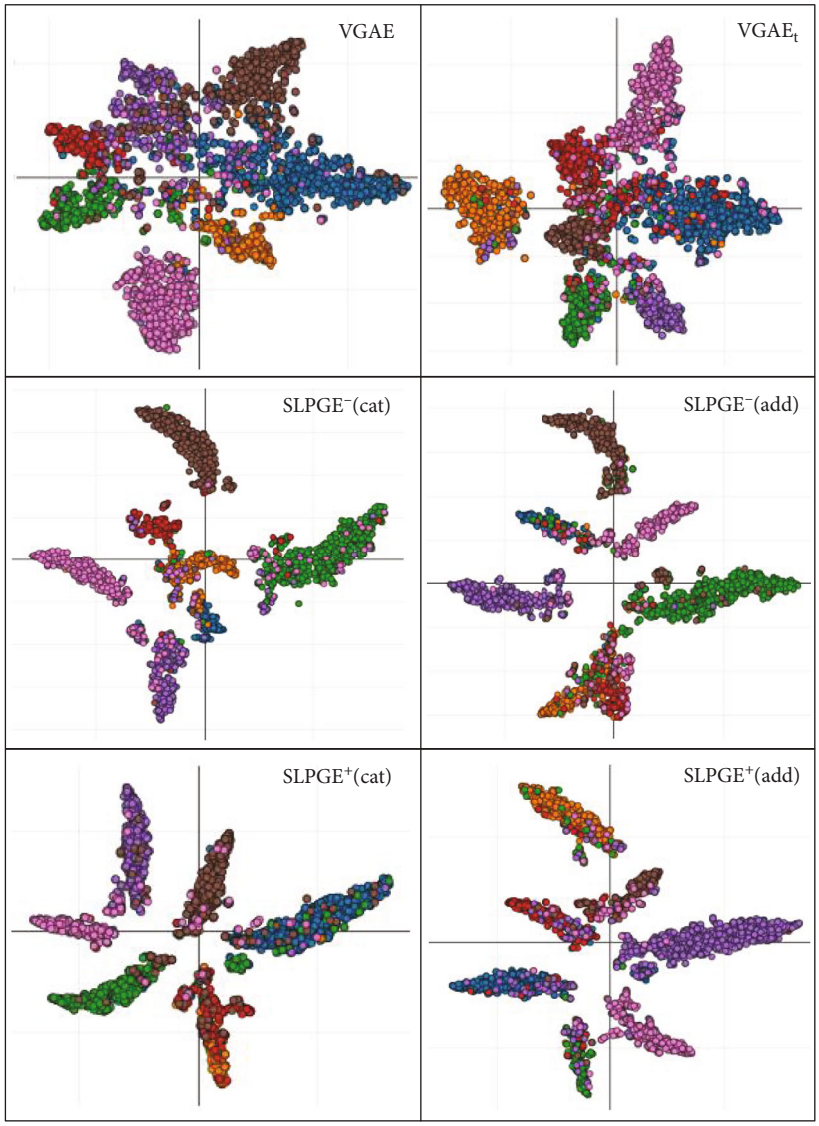

Figure 6: Visualization of node classification on Cora.

6. Simulation

In this section, we will evaluate the performance of SLPGE on two public datasets, Cora [35] and Yale [14].

6.1. Experiment Setting

(1) *Datasets*. Cora is a citation network composed of 7 categories of machine learning papers. Cora includes 2708 papers as **V** and 5278 citation relationships between papers as **E**. 1433 unique words appear in all papers as the attributes of **V**. Yale is a social network including 8578 people and 188 attributes. The class year attribute divides the nodes into 7 categories. Part of links and labels of the datasets are used as training sets.

(2) *Training*. The experimental parameters are shown in Table 2. The initial features of nodes are 1433 and 188 dimensions. $\mathbf{Z}_f$ and $\mathbf{Z}_p$ are both 8-dim in Cora and 7-dim in Yale. As shown in Figure 5, we have two splicing modes of $\mathbf{Z}_f$ and $\mathbf{Z}_p$ in SLPGE: "concatenate (cat)" and "add," where "cat" means stacking $\mathbf{Z}_f$ and $\mathbf{Z}_p$ in the horizontal direction (i.e., column order) and "add" means that the elements in $\mathbf{Z}_f$ and $\mathbf{Z}_p$ are added correspondingly. $\mathbf{Z}$ is 16-dim and 14-dim when using "cat" and 8-dim and 7-dim when using "add." The embeddings of two nodes in $\mathbf{Z}_f$ are concatenated together as an edge embedding, so the dimension of edge embedding is twice as large as node embedding.

Besides, we take the original graph $\mathbf{G}$ and the training graph $\mathbf{G}_t$ with sensitive links deleted as the input of VGAE to compare with SLPGE. At the same time, we use TSNE to visualize $\mathbf{Z}_f$ in 2-dim to observe the node classification result, and the nodes belonging to the same label are

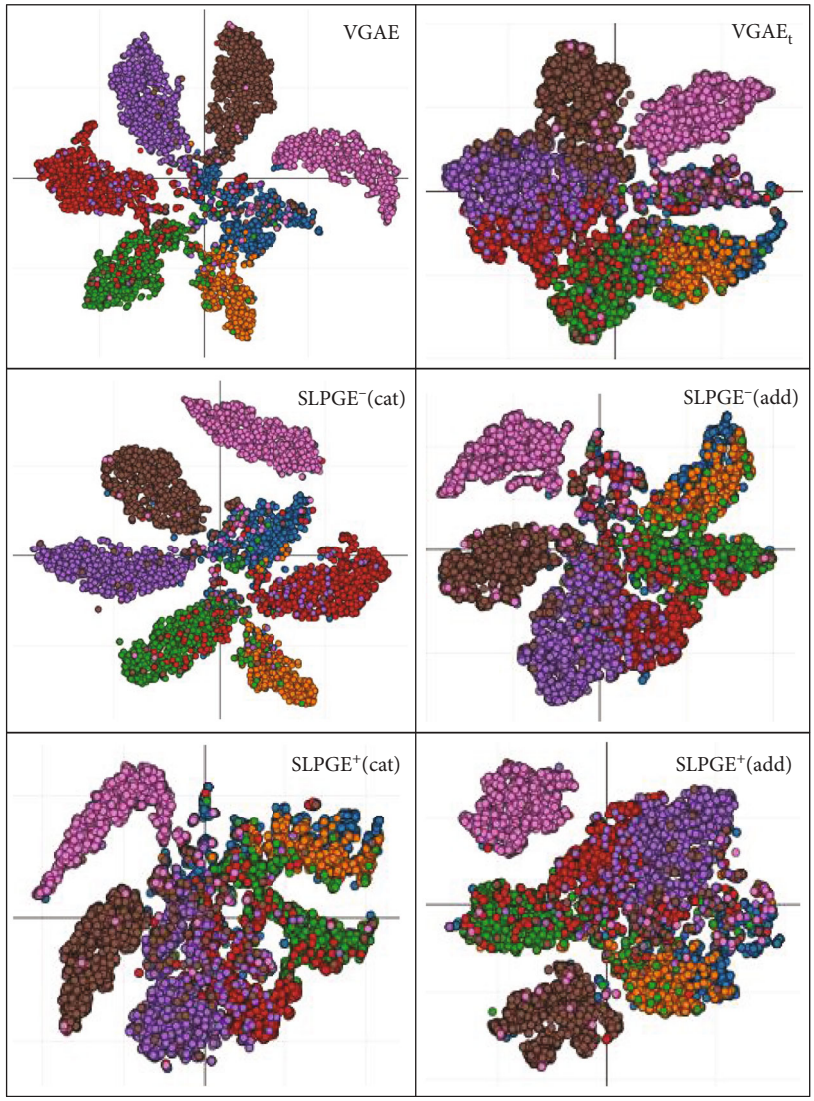

Figure 7: Visualization of node classification on Yale.

represented by the same color. In essence, TSNE uses PCA to reduce the dimension of the feature and then maps it to a 2-dimensional or 3-dimensional space for visualization to observe each layer's feature distribution.

(3) *Attack*. 100 and 200 edges with larger node degrees in the training sets are selected as the sensitive links of Cora and Yale, respectively, and m is 10 and 15 in Algorithm 1. We randomly select 200 nonsensitive links and 200 nonexistent links to form $\mathbf{E}_{\text{know}}$ which has been exposed to the attackers. Moreover, the edge embeddings of $\mathbf{E}_{\text{know}}$ will be used as the training set to train the attack models. The edge embeddings of the same number of sensitive and nonsensitive links are the input of the attack models. We train each model four times, and the attack models make 10 predictions after each training. Finally, the averages of the 40 prediction results are taken as the prediction accuracy of sensitive and nonsensitive links.

6.2. Result Analysis. We carried out our experiments under four models: **VGAE**, **VGAE$_t$**, **SLPGE$^+$**, and **SLPGE$^-$**. **VGAE** means the input is the original graph without any modification. **VGAE$_t$** means the input is the training graph in which the sensitive links are deleted. Our SLPGE is divided into two types: **SLPGE$^+$** and **SLPGE$^-$** where $\mathbf{Z}_p$ comes from Algorithms 1 and 2, respectively.

Figures 6 and 7 show node classification of SVM under different models for Cora and Yale in a visualization method, respectively. In each subgraph, the points in the same color constitute a cluster, representing different classes. A larger distance between different clusters means higher accuracy. Corresponding numerical results are listed in Tables 3 and 4. The decline degree of five indicators of **VGAE$_t$**, **SLPGE$^+$**,

Table 3: The results on Cora.

Model	Splicing mode	SVM		MLP		Reconstruction		
		Acc_{sl}	Acc_{nsl}	Acc_{sl}	Acc_{nsl}	Acc_{recon}	Re c_{recon}	Acc_{node}
VGAE	—	87.5	85.0	84.1	85.0	88.6	90.7	83.5
$VGAE_t$	—	75.2	83.0	80.4	83.5	84.7	89.7	79.9
$SLPGE^-$	cat	66.5	80.1	65.8	79.1	85.6	86.8	76.4
	add	61.6	84.6	61.3	80.5	83.5	85.4	74.1
$SLPGE^+$	cat	68.0	82.0	61.8	79.8	86.6	87.3	74.5
	add	61.2	82.3	60.6	84.4	84.4	87.2	77.2

Table 4: The results on Yale.

Model	Splicing mode	SVM		MLP		Reconstruction		
		Acc_{sl}	Acc_{nsl}	Acc_{sl}	Acc_{nsl}	Acc_{recon}	Re c_{recon}	Acc_{node}
VGAE	—	84.5	84.5	76.5	77.0	72.1	84.5	81.0
$VGAE_t$	—	81.0	83.0	74.0	73.5	73.1	85.7	77.6
$SLPGE^-$	cat	75.0	81.2	65.0	71.0	67.3	79.7	80.1
	add	74.5	84.5	65.2	71.7	65.6	71.4	76.5
$SLPGE^+$	cat	76.5	83.6	65.8	68.1	68.9	75.8	75.6
	add	73.7	82.0	67.5	71.5	68.8	75.1	75.5

Table 5: The decline degree of five indicators compared with VGAE on Cora.

Model	Splicing mode	SVM		MLP		Reconstruction		
		Acc_{sl}	Acc_{nsl}	Acc_{sl}	Acc_{nsl}	Acc_{recon}	Re c_{recon}	Acc_{node}
$VGAE_t$	—	14.05	2.35	4.40	1.76	4.40	1.10	4.31
$SLPGE^-$	cat	24.00	5.76	21.76	6.94	3.39	4.30	8.50
	add	29.60	0.47	27.11	5.29	5.76	5.84	11.26
$SLPGE^+$	cat	22.28	3.52	26.52	6.12	2.26	3.75	10.78
	add	30.05	3.17	27.94	0.71	4.74	3.86	7.54

Table 6: The decline degree of five indicators compared with VGAE on Yale.

Model	Splicing mode	SVM		MLP		Reconstruction		
		Acc_{sl}	Acc_{nsl}	Acc_{sl}	Acc_{nsl}	Acc_{recon}	Re c_{recon}	Acc_{node}
$VGAE_t$	—	4.14	1.78	3.27	4.55	-1.39	-1.42	4.20
$SLPGE^-$	cat	11.24	3.91	15.03	7.79	6.66	5.68	1.11
	add	11.83	0.00	14.77	6.88	9.02	15.50	5.56
$SLPGE^+$	cat	9.47	1.07	13.99	11.56	4.44	10.30	6.67
	add	12.78	2.96	11.76	7.14	4.58	11.12	6.79

and **SLPGE**$^-$ compared with **VGAE** is shown in Tables 5 and 6, and the decline degree are calculated by $|A - B|/B\%$, where B represents **VGAE** and A represents the others.

6.2.1. Privacy. There is a comparison of Acc_{sl} of the four models in Tables 3 and 4 that Acc_{sl} of $\mathbf{VGAE_t}$, **SLPGE**$^+$, and **SLPGE**$^-$ decrease in varying degrees, but Acc_{sl} of **SLPGE**$^+$ and **SLPGE**$^-$ decrease more. Especially for Cora, **SLPGE**$^+$ reduces Acc_{sl} by 30.05% at most and 22.28% at least on the basis of **VGAE** while $\mathbf{VGAE_t}$ reduces Acc_{sl} by 14.05% at most. For Yale, **SLPGE**$^+$ reduces Acc_{sl} by 15.03% at most and 9.46% at least on the basis of **VGAE** while $\mathbf{VGAE_t}$ reduces Acc_{sl} by 4.14% at most. The privacy of SLPGE has significant improvement compared with $\mathbf{VGAE_t}$.

Although the privacy of SLPGE is 1.3 ~ 3.6 times higher than that of $\mathbf{VGAE_t}$ on Yale, the protection effect of sensitive links on Yale is not as good as that on Cora, which results from the fact that the node attributes of Yale are more closely related

to the links. This also signifies that similar attributes will make the privacy information between nodes more difficult to remove. In general, these comparisons can confirm that our SLPGE has better performance on sensitive link protection.

6.2.2. Utility. The loss of partial utility is the necessary cost of privacy protection. Taking **VGAE** as a comparison, we can see that the classification accuracy of four variant models has decreased, reflecting a partial sacrifice of data utility. $\mathrm{Acc}_{\mathrm{nsl}}$, $\mathrm{Acc}_{\mathrm{link}}$, Re $\mathrm{c}_{\mathrm{recon}}$, and $\mathrm{Acc}_{\mathrm{node}}$ of SLPGE and $\mathbf{VGAE_t}$ all decrease simultaneously, but the decline ranges are generally lower than that of $\mathrm{Acc}_{\mathrm{sl}}$. From Tables 5 and 6, it can be seen that $\mathrm{Acc}_{\mathrm{nsl}}$ of SLPGE decrease by 6.94% and 11.56% at most on the basis of **VGAE** for Cora and Yale, but the two $\mathrm{Acc}_{\mathrm{sl}}$ decrease more, reaching 21.76% and 13.99%. $\mathrm{Acc}_{\mathrm{link}}$ of SLPGE decrease by 5.75% and 9.07% at most for Cora and Yale. The maximum decline ranges of $\mathrm{Acc}_{\mathrm{link}}$, Re $\mathrm{c}_{\mathrm{recon}}$, and $\mathrm{Acc}_{\mathrm{node}}$ of SLPGE on two datasets are 5.76% and 9.02%, 5.84% and 15.50%, and 11.26% and 6.79%, which are basically lower than the decline ranges of $\mathrm{Acc}_{\mathrm{sl}}$. Tables 5 and 6 reflect the tradeoff between privacy and utility.

6.2.3. Models. The data in four tables show that $\mathbf{SLPGE}^{+}$ and $\mathbf{SLPGE}^{-}$ are very close in performance on privacy and utility, which also proves that both Algorithms 1 and 2 are feasible. For the two splicing modes, the privacy and utility of mode "add" are better than those of mode "cat." The analysis of this result is as follows.

The distributions of $\mathbf{Z}_f$ and $\mathbf{Z}_p$ both approach $\mathcal{N}(0,1)$ (standard normal distribution), and the weight of privacy information in $\mathbf{Z}_p$ is large and fixed. When $\mathbf{Z}$ obtained by adding $\mathbf{Z}_f$ and $\mathbf{Z}_p$ is to fit the link labels of the original graph after decoding, the MSE loss function will force $\mathbf{Z}_f$ to reduce the weight of privacy information, so that we can squeeze more privacy information. Therefore, the combination of MSE and mode "add" is better.

Overall, our SLPGE reduces the prediction accuracy of sensitive links to varying degrees, from which we can conclude that our model is effective. While protecting the privacy of sensitive links, some utility will be sacrificed, which may be structure information or attribute information. From the result analysis, it can be confirmed that SLPGE can retain most of the utility. In practical application, part of the structure of the model can be adjusted to meet different task requirements.

7. Conclusion

The problems of individual privacy under the interconnection of all things are ubiquitous. The research on link protection against link prediction in IoT is of great significance for entity privacy. Through the simulation of the datasets, the feasibility of our SLPGE is preliminarily verified. However, multifaceted challenges remain in the research on link protection. Our datasets are just static graphs, in which the nodes belong to different categories at the same level, and the edges only represent reference and social relationships. In heterogeneous scenarios, nodes can be of different levels, edges between the nodes may have diverse meanings, and the weight of the edges are no longer all equal to one. The weight of edges reflects the difference in the degree of communication between nodes.

Furthermore, in dynamic graphs, the entry and exit of nodes will affect the graph structure and the privacy information of sensitive links in real time. The attackers can collect more information for inference attacks. The greatest challenge is that the researches on resisting graph disturbance and enhancing the robustness of link prediction continue to emerge, which increases the difficulty of sensitive link protection. Therefore, we will emphasize the sensitive link protection in weighted graphs and dynamic graphs in our follow-up research.

Appendix

In Section 5.2, we use the two modes of "concatenate (cat)" and "add" to combine $\mathbf{Z}_f$ and $\mathbf{Z}_p$. The following is an explanation of these two operations. "cat" and "add" are two splicing modes of $\mathbf{Z}_f$ and $\mathbf{Z}_p$ in SLPGE. "cat" means stacking $\mathbf{Z}_f$ and $\mathbf{Z}_p$ in the horizontal direction, and "add" means that the elements in $\mathbf{Z}_f$ and $\mathbf{Z}_p$ are added correspondingly. Here, we will intuitively show how to get $\mathbf{Z}(\mathrm{cat})$ and $\mathbf{Z}(\mathrm{add})$ and explain their meanings. We assume that both $\mathbf{Z}_f$ and $\mathbf{Z}_p$ are 3 times 2 matrices,

$$\mathbf{Z}_f = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \\ c_1 & c_3 \end{bmatrix}, \quad \mathbf{Z}_p = \begin{bmatrix} a_3 & a_4 \\ b_3 & b_4 \\ c_3 & c_4 \end{bmatrix}, \tag{A.1}$$

then

$$\mathbf{Z}(\mathrm{cat}) = \left[\mathbf{Z}_f \mathbf{Z}_p\right] = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \end{bmatrix}, \quad \mathbf{Z}(\mathrm{add}) = \mathbf{Z}_f + \mathbf{Z}_p = \begin{bmatrix} a_1 + a_3 & a_2 + a_4 \\ b_1 + b_3 & b_2 + b_4 \\ c_1 + c_3 & c_2 + c_4 \end{bmatrix}. \tag{A.2}$$

In Part II, the reconstructed adjacency matrix $\widehat{\mathbf{A}}$ is obtained by the inner product of $\mathbf{Z}(\mathbf{Z}(\mathrm{cat})\ \mathrm{or}\ \mathbf{Z}(\mathrm{add}))$, i.e., $\widehat{\mathbf{A}}(\mathrm{cat}) = \mathbf{Z}(\mathrm{cat})\mathbf{Z}(\mathrm{cat})^T$ and $\mathbf{A}(\mathrm{add}) = \mathbf{Z}(\mathrm{add})\mathbf{Z}(\mathrm{add})^T$ whose detailed calculations are shown in the bottom.

We can see that $\widehat{\mathbf{A}}(\mathrm{add}) = \widehat{\mathbf{A}}(\mathrm{cat}) + \mathbf{B}$, and $\mathbf{B}$ is a cross-multiplying term matrix. Since $\mathbf{Z}_p$ is fixed, $(a_3, a_4, b_3, b_4, c_3$, and c_4 is fixed), the loss function will force Z_f to constantly

adjust so that $\widehat{\mathbf{A}}$ is close to $\mathbf{A}$. Because $\widehat{\mathbf{A}}(\text{add})$ has more cross-multiplying terms, $\widehat{\mathbf{A}}(\text{add})$ may exert greater pressure $\mathbf{Z}_f(a_1, a_2, b_1, b_2, c_1, \text{and } c_2)$. Based on the above analysis, we chose these two modes to get $\mathbf{Z}$:

$$\widehat{\mathbf{A}}(\text{cat}) = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \end{bmatrix} \times \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \\ a_4 & b_4 & c_4 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{4} a_i^2 & \sum_{i=1}^{4} a_i b_i & \sum_{i=1}^{4} a_i c_i \\ \sum_{i=1}^{4} b_i a_i & \sum_{i=1}^{4} b_i^2 & \sum_{i=1}^{4} b_i c_i \\ \sum_{i=1}^{4} c_i a_i & \sum_{i=1}^{4} c_i b_i & \sum_{i=1}^{4} c_i^2 \end{bmatrix},$$

$$\widehat{\mathbf{A}}(\text{add}) = \begin{bmatrix} a_1 + a_3 & a_2 + a_4 \\ b_1 + b_3 & b_2 + b_4 \\ c_1 + c_3 & c_2 + c_4 \end{bmatrix} \times \begin{bmatrix} a_1 + a_3 & b_1 + b_3 & c_1 + c_3 \\ a_2 + a_4 & b_2 + b_4 & c_2 + c_4 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{4} a_i^2 & \sum_{i=1}^{4} a_i b_i & \sum_{i=1}^{4} a_i c_i \\ \sum_{i=1}^{4} b_i a_i & \sum_{i=1}^{4} b_i^2 & \sum_{i=1}^{4} b_i c_i \\ \sum_{i=1}^{4} c_i a_i & \sum_{i=1}^{4} c_i b_i & \sum_{i=1}^{4} c_i^2 \end{bmatrix} + \begin{bmatrix} 2a_1a_3 + 2a_2a_4 & a_1b_3 + a_3b_1 + a_2b_4 + a_4b_2 & a_1c_3 + a_3c_1 + a_2c_4 + a_4c_2 \\ b_1a_3 + b_3a_1 + b_2a_4 + b_4a_2 & 2b_1b_3 + 2b_2b_4 & b_1c_3 + b_3c_1 + b_2c_4 + b_4c_2 \\ c_1a_3 + c_3a_1 + c_2a_4 + c_4a_2 & c_1b_3 + c_3b_1 + c_2b_4 + c_4b_2 & 2c_1c_3 + 2c_2c_4 \end{bmatrix}. \tag{A.3}$$

Data Availability

Cora [35] is a citation network composed of 7 categories of machine learning papers. Cora includes 2708 papers and 5278 citation relationships between papers. 1433 unique words appear in all papers as the attributes. Yale is a social network including 8578 people and 188 attributes. The class year attribute divides the nodes into 7 categories. Part of links and labels of the datasets are used as training sets. K. Li, "The data about the facebook friendships of yale university." [Online]. Available: https://github.com/KaiyangLi1992/Privacy-Preserving-Social-Network-Embedding.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant 2019JBZ001, in part by the Beijing Natural Science Foundation under Grant 4202054, and in part by the National Natural Science Foundation of China under Grant 61871023 and Grant 61931001.

References

[1] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Transactions on Network Science & Engineering*, vol. 7, no. 2, pp. 766–775, 2018.

[2] H. Yang, Y. Liang, J. Yuan, Q. Yao, and J. Zhang, "Distributed blockchain-based trusted multi-domain collaboration for mobile edge computing in 5g and beyond," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 11, 2020.

[3] Z. Cai and Z. He, "Trading private range counting over big iot data," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, Dallas, TX, USA, 2019.

[4] X. Zheng and Z. Cai, "Privacy-preserved data sharing towards multiple parties in industrial iots," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 968–979, 2020.

[5] Z. Cai, Z. He, X. Guan, and Y. Li, "Collective data-sanitization for preventing sensitive information inference attacks in social

networks," *IEEE Transactions on Dependable & Secure Computing*, vol. 15, no. 4, pp. 577–590, 2016.

[6] K. Liu and E. Terzi, "Towards identity anonymization on graphs," in *in Acm Sigmod International Conference on Management of Data*, Vancouver, Canada, 2008.

[7] F. O. Rousseau, J. Casas Roma, and M. Vazirgiannis, "Community-preserving anonymization of graphs," *Knowledge & Information Systems*, vol. 54, no. 2, pp. 315–343, 2018.

[8] A. Milani Fard and K. Wang, "Neighborhood randomization for link privacy in social network analysis," *World Wide Web internet & Web Information Systems*, vol. 18, no. 1, pp. 9–32, 2015.

[9] P. Mittal, C. Papamanthou, and D. Song, "Preserving link privacy in social network based systems," *Computer Science*, 2012, https://arxiv.org/abs/1208.6189.

[10] K. Zhou, T. P. Michalak, T. Rahwan, M. Waniek, and Y. Vorobeychik, "Attacking similarity-based link prediction in social networks," 2018, https://arxiv.org/abs/1809.08368.

[11] J. Chen, Z. Shi, Y. Wu, X. Xu, and H. Zheng, "Link prediction adversarial attack," 2018, https://arxiv.org/abs/1810.01110.

[12] S. Yu, M. Zhao, C. Fu et al., "Target defense against link-prediction-based attacks via evolutionary perturbations," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 2, pp. 1–767, 2019.

[13] M. Waniek, K. Zhou, Y. Vorobeychik, E. Moro, and T. Rahwan, "How to hide one's relationships from link prediction algorithms," *Scientific Reports*, vol. 9, no. 1, p. 12208, 2019.

[14] K. Li, G. Luo, Y. Ye, W. Li, S. Ji, and Z. Cai, "Adversarial privacy preserving graph embedding against inference attack," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6904–6915, 2021.

[15] L. Sweeney, "k-anonymity: a model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[16] X. Ying and X. Wu, "Randomizing social networks: a spectrum preserving approach," in *in Proceedings of the SIAM International Conference on Data Mining, SDM 2008*, Atlanta, Georgia, USA, 2008.

[17] Y. Li, H. Shen, C. Lang, and H. Dong, "Practical anonymity models on protecting private weighted graphs," *Neurocomputing*, vol. 218, pp. 359–370, 2016.

[18] M. Yuan, C. Lei, P. S. Yu, and T. Yu, "Protecting sensitive labels in social network data anonymization," *IEEE Transactions on Knowledge & Data Engineering*, vol. 25, no. 3, pp. 633–647, 2013.

[19] S. Chester and G. Srivastava, "Social network privacy for attribute disclosure attacks," in *International Conference on Advances in Social Networks Analysis & Mining*, Kaohsiung, Taiwan, 2011.

[20] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava, "Anonymizing Social Networks," in *34th International Conference on Very large Data Bases (VLDB)*, Auckland, New Zealand, 2008.

[21] L. Liu, J. Wang, J. Liu, and J. Zhang, "Privacy preservation in social networks with sensitive edge weights," in *Siam International Conference on Data Mining*, Sparks, Nevada, USA, 2009.

[22] E. Zheleva and L. Getoor, "Preserving the privacy of sensitive relationships in graph data," *International Journal of Computer Trends & Technology*, vol. 17, no. 1, 2014.

[23] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems*, vol. 13, no. 4, pp. 18–28, 1998.

[24] H. L. Jensen, "Using neural networks for credit scoring," *Managerial Finance*, vol. 18, no. 6, pp. 15–26, 1992.

[25] S. Yang, H. Jian, Z. Ding, H. Zha, and C. L. Giles, "Iknn: informative k-nearest neighbor pattern classification," in *European Conference on Knowledge Discovery in Databases: Pkdd*, Berlin, Heidelberg, 2007.

[26] J. Gu, Z. Wang, J. Kuen, L. Ma, and G. Wang, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018.

[27] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks," 2017, https://arxiv.org/abs/1801.01078.

[28] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," 2013, https://arxiv.org/abs/1312.6203.

[29] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, https://arxiv.org/abs/1609.02907.

[30] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016, https://arxiv.org/abs/1611.07308.

[31] D. J. Im, S. Ahn, R. Memisevic, and Y. Bengio, "Auto-encoding variational Bayes," https://arxiv.org/abs/1312.6114.

[32] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *Twenty-Seventh International Joint Conference on Artificial Intelligence IJCAI-18*, 2018https://arxiv.org/abs/1802.04407.

[33] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., *Generative adversarial nets*, MIT Press, 2014.

[34] Z. Cai, Z. Xiong, H. Xu, P. Wang, and Y. Pan, "Generative adversarial networks," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–38, 2021.

[35] S. Prithviraj, N. Galileo, B. Mustafa, G. Lise, and G. Brian, *Collective Classification in Network Data*, Ai Magazine, 2008.