PAPER Special Section on Trust, Security and Privacy in Computing and Communication Systems

A Privacy Protection Method for Social Network Data against Content/Degree Attacks*

Min Kyoung SUNG[†], **Ki Yong LEE**[††], **Jun-Bum SHIN**[†††], ***Nonmembers*, *and* Yon Dohn CHUNG**[†a)], ***Member***

SUMMARY Recently, social network services are rapidly growing and this trend is expected to continue in the future. Social network data can be published for various purposes such as statistical analysis and population studies. When social network data are published, however, the privacy of some people may be disclosed. The most straightforward manner to preserve privacy in social network data is to remove the identifiers of persons from the social network data. However, an adversary can infer the identity of a person in the social network by using his/her background knowledge, which consists of content information such as the age, sex, or address of the person and structural information such as the number of persons having a relationship with the person. In this paper, we propose a privacy protection method for social network data. The proposed method anonymizes social network data to prevent privacy attacks that use both content and structural information, while minimizing the information loss or distortion of the anonymized social network data. Through extensive experiments, we verify the effectiveness and applicability of the proposed method.

key words: *privacy, social network, data publication, k-anonymity*

1. Introduction

With the rapid increase of social network services, such as Facebook (facebook.com), Buzz (buzz.com), LinkedIn (linked.com), and Twitter (twitter.com), many researchers have paid much attention to social networks in recent years. These kinds of applications provide services based on the personal information of people and relationships among them. Researchers can analyze these social network data for the study of marketing, epidemiology, and so on. However, a social network may contain some sensitive information that discloses privacy of individuals [3], which causes social problems. For example, there are some personal information that is too sensitive to be disclosed, such as salary, disease, and credit rating. Moreover, a person can be the target of a crime that invades the person's privacy. A simple way to preserve the privacy of persons in social network data is to remove their identifiers (e.g., name or social security number (SSN)) from the social network data. However, even when the identifiers of persons are removed, there is still a risk of the privacy disclosure, as will be demonstrated in the following example.

Example 1. Let G be a social network (Fig. 1). Each node n_i ($i = 1, \cdots, 6$) represents a person and each edge represents a relationship between persons. Each node contains content information, i.e., age, sex, and salary of a person. Note that the identifier of each person is not presented in G. Here, age and sex are non-sensitive information, whereas salary is sensitive one. Suppose an adversary knows that Tom is a 26 year old male and the number of persons having a relationship with him is 4. Then, the adversary infers that n_4 corresponds to Tom. Although the identifier of Tom is not presented in G, the adversary knows that the salary of Tom is $ 40,000. □

The above example shows that even when the identifiers of persons are removed from social network data, an adversary can disclose the privacy of a person by combining his/her background knowledge with the published social network data. Therefore, a data holder has to anonymize social network data before publishing them.

In general, a social network is represented as a graph, where each node represents a person and each edge represents a relationship between persons as described in the above example. The graph also contains content information, which corresponds to the values stored in each node. In order to achieve the anonymization of social network data, nodes and edges in the social network can be modified, added, deleted, or clustered. Several methods have been proposed to preserve the privacy of social networks in the past [6], [7], [9], [13], [15], [17], [18]. However, most of them considered only structural information [7], [9], [17], [18]. Some recent work [6], [13], [15] considered both structural information and content information, but have some limitations as described below.

Campan et al. [6] suggested a node clustering method that does not add or delete edges, but their method causes the loss of detailed structural information such as a relationship

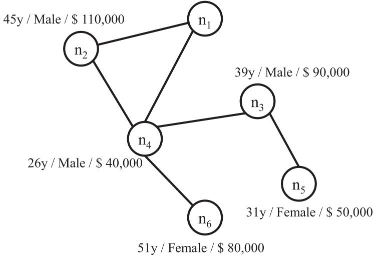


Fig. 1 An example of social network G.

Manuscript received March 15, 2011.

[†] The authors are with the Department of Computer Science and Engineering, College of Information and Communication, Korea University, Seoul, Korea.

[††] The author is with the Department of Computer Science, Sookmyung Women's University, Seoul, Korea.

[†††] The author is with DMC R&D, Samsung Electronics, Co., Ltd., Korea.

[*] This work was supported by Samsung Electronics.

a) E-mail: ydchung@korea.ac.kr (Corresponding author)

DOI: 10.1587/transinf.E95.D.152

between two specific persons. Zheleva et al. [15] proposed an edge clustering method which considers the content information attached to edges, but this method also causes the same problem as [5]. Wei et al. [13] suggested a method that produces unconnected social networks. However, unconnected social networks cause unnecessary confusion to a data receiver when he/she tries to reconstruct the original social network from the unconnected social networks. Furthermore, it considers only limited types of content information.

In this paper, we propose a novel privacy preservation method for social network data. For a given integer parameter k and social network G, the proposed method modifies G to an anonymized social network G_A by generalizing content information and adding extra edges to prevent privacy disclosure. As the structural information, we consider only the degree of each node (i.e., the number of edges connected to the node) and leave other types of structural information (e.g., sub-graphs) as future work because it complicates the problem more than necessary. In the anonymized social network G_A, every node has at least (k - 1) indistinguishable nodes that have the same degree and content information. Thus, an adversary who knows the degree or content information of the target victim cannot correctly identify the target victim from G_A with more than $1/k$ probability. While preserving the privacy in the social network, our method minimizes the information loss caused by edge-addition and content-generalization as much as possible. The contributions of our paper are summarized as follows:

- We define information loss metrics that measure the loss of information incurred in social network anonymization.
- We propose a social network data anonymizaion method that can conserve the whole structural information as much as possible. That is, the proposed method can conserve more structural information than the previous methods [6], [13].
- Unlike the previous privacy preserving methods for the social network that consider only structural information [15], we consider both of the structural information and the content information of nodes in the social network, which contains a variety of person-specific information.
- Whereas the previous privacy preserving methods produce a number of unconnected social networks [13], our method produces a single connected social network.
- Through extensive experiments, we show that the proposed method performs better than the previous methods with respect to the level of privacy preservation and the loss of information.

The remainder of this paper is organized as follows. Section 2 introduces the previous privacy preserving techniques in relational databases and social networks. In Sect. 3, we formulate the data/attack models and information loss metrics we use in the paper. In Sect. 4, we describe our anonymization algorithm. We show our experimental results in Sect. 5, and conclude the paper in Sect. 6.

2. Related Work

When publishing data, a data holder such as hospitals has to anonymize the data to preserve the privacy of persons contained in the data. Generalization and suppression are popular techniques to anonymize the data. In those techniques, the original data are modified and/or changed to satisfy the required level of privacy protection. When anonymizing the data, we have to consider the following two factors: (1) the level of privacy preservation supported by the anonymization method and (2) the amount of information loss or distortion caused by the data anynymization.

Many researchers have worked for privacy preservation within the relational data [10], [11], [14], [16]. Sweeney et al. [11] pointed out that it is not sufficient for preserving the privacy in the published data to remove the identifiers of persons such as name or SSN, and proposed the notion of k-anonymity. The constraint of the k-anonymity indicates that every record in the published data must have at least (k-1) other indistinguishable records. Machanavajjhala et al. [10] presented a model of ℓ-diversity which complements the shortcoming of k-anonymity, i.e., the lack of diversity in sensitive values. The ℓ-diversity requires that (1) the data satisfy the k-anonymity and also (2) the data have ℓ different sensitive values among k indistinguishable records. There were some considerations on dynamic data that is periodically updated [14], [16], where the data holder should ensure that the privacy is not disclosed by combining the previously released data and the current ones. In the data mining field, there was a study that attempts to maintain privacy while extracting useful information from databases [12].

However, although the above methods are suitable for relational data, they are not appropriate for social network data represented by graphs. For this reason, many researchers have studied methods for privacy preservation of the social network. The privacy leakage problems on social networks are classified as follows:

- **Sub-graph Attacks (Neighborhood Attacks):** Zou et al. [17] proposed an anonymization method to prevent structural attacks in the social network. They considered a sub-graph attack, called the d-neighborhood attack, where an adversary can identify a target victim in the social network by using his/her background knowledge, which is represented as a specific sub-graph. To preserve the privacy against such attacks, they modified the original social network so that the anonymized social network could contain at least k isomorphic sub-graphs.

- **Degree Attacks:** An adversary who knows the degree of a specific target victim in the social networks can acquire the sensitive data in the social network. The notion of k-degree anonymity [9] is proposed to prevent such attacks where every node in the social network should have at least (k-1) indistinguishable other nodes by its degree. That is, an adversary who knows the degree of a target victim cannot identify the victim with more than $1/k$ probability. The work [9] presented a simple and efficient edge-addition algorithm to prevent such attacks.

- **Sub-graph + Degree Attacks:** There can be an adversary who knows both sub-graphs and the degree of the target victim. Hay et al. [7] considered such an adversary and proposed a method that produces at least k candidate answers to any structural queries, namely k-candidate anonymity. Zou et al. [18] further observed that an adversary can perform different structural queries to re-identify a specific person. They proposed an edge-addition approach since the previous approach [7] incurs too much loss of the structural information. They also considered the dynamic release of the social network data.

The above methods assume that adversaries know only the structural information such as sub-graphs or degrees of nodes. However, there can be adversaries who know both the structural information and the content information of the social network. Some recent work [6], [13], [15] dealt with this kind of privacy attacks which are more practical in the real world.

- **Node/Edge-Clustering Approach:** The anonymization approach proposed in [15] considered the content information of edges in the social networks. It uses an edge-clustering approach to hide the content information of edges. The content information of nodes was also considered by Campan et al. [6], where a node-clustering approach was used. The node-clustering approach makes super-nodes, by gathering nodes that have a similar structural and content information. That is, a super-node consists of several similar nodes.

- **Edge-Addition/Deletion Approach:** Wei et al. [13] presented an edge-addition/deletion approach. Their method produces an anonymized social network in three phases. In the creation phase, it creates a number of k sub-graphs from the original social network and generalizes the node labels such that every node in each of k sub-graphs has the same labels. In the perturbation phase, it adds/deletes edges to make every node in each sub-graph have the same structural information. In the construction phase, it inserts connectivity information among k sub-graphs.

As discussed in [18], the clustering approaches [6], [15] produce an anonymized social network that contains only a summary of structural information about the original social network. That is, the published social network does not contain detailed information about the relationship between persons. On the other hand, Wei et al. [13] used an edge-addition/deletion approach, but it produces unconnected social networks after anonymization, even when the original social network is a connected one.

3. Problem Formulation and Information Loss Metrics

3.1 The Data Model

The social network can be represented as a graph consisting of nodes and edges. We assume here that the social network is expressed as a simple graph $G = \{N, E\}$, where N is a set of nodes and $E \subseteq N \times N$ is a set of edges. A simple graph is an undirected graph where there are no loops and no multi-edges between nodes. A node corresponds to a person and an edge corresponds to a relationship between persons. We assume that one person corresponds to one node. That is, a person appears only once in the social network. For convenience, we interchangeably use two terms 'graph' and 'network' in the paper.

Each node in the social network contains content information such as the age, disease and sex of a person. The content information consists of a set of attributes, which are classified into three groups: *IDentifier (ID)*, *Quasi-Identifier (QI)* and *Sensitive Attribute (SA)*. *ID* is the set of attributes that explicitly express the identity of a person, such as 'name' or 'SSN'. *QI* is the set of attributes that do not explicitly express personal identify information, but could potentially disclose the privacy when combined with the background knowledge of adversaries. For example, 'age' and 'sex' are *QI* attributes. *SA* is the set of attributes that contain sensitive information that must not be disclosed, such as disease and salary.

3.2 The Attack Model

It is difficult to estimate how much background knowledge adversaries have. If adversaries have a lot of background knowledge, we have to apply a stricter privacy preserving method. Otherwise, we can apply a relaxed privacy preserving method. In this paper, we assume that adversaries have the following background knowledge:

- Adversaries know whether the target victim exists in the published social network.

- Adversaries know the *QI* attribute values of the target victim. (i.e., the content information)

- Adversaries know the degree of the target victim (i.e., the number of nodes connected to the target victim) in the social network.

In other words, adversaries are assumed to have background knowledge with respect to both the content and structural information of the target victim.

Definition 1. (Privacy disclosure problem)
We call it a privacy disclosure that if adversaries can infer the sensitive information of the target victim in probability of more than $1/k$ by using the published social network and adversaries' background knowledge.

3.3 Overview of the Proposed Method

To prevent a privacy attack described in *Definition 1*, each node in the social network should have at least (k - 1) other nodes indistinguishable from it. This guarantees that an adversary cannot uniquely identify a node with a probability of more than $1/k$. In order to ensure this, the proposed method partitions the social network into groups, each of which has at least k nodes, and makes nodes in each group have the same QI attribute values and degree. We call each group an *Equivalence Class (EC)*. To make nodes in each EC have the same QI attribute values, the proposed method modifies the QI attribute values of nodes in the same EC to reflect their common concepts. For example, if two nodes in the same EC have QI attribute values of 'Germany' and 'U.K.', they can be modified to 'Europe'. To make nodes in each EC have the same degree, we can use many strategies, such as node-addition/deletion, and edge-addition/deletion. In this paper, we use an edge-addition strategy because it distorts structural information less than the others. For example, a node-deletion strategy deletes not only a node but also edges connected to the node. More details are described in the next Sect. 3.4. Because each EC has at least k indistinguishable nodes, it is clear that there are always at least k nodes that correspond to an adversary's background knowledge. Therefore, a node cannot be uniquely identified with a probability of more than $1/k$.

We now more formally describe the proposed method. Let G be a social network. For a given integer parameter k and G, the proposed method produces an anonymized version of G, denoted by G_A, by modifying the QI attribute values and degree of nodes in G. To produce G_A from G, the proposed method partitions nodes in G into equivalence classes $EC_1, \cdots, EC_q$ such that $EC_i \cap EC_j = \emptyset$ ($1 \leq i < j \leq q$) and $|EC_i| \geq k$, where $|EC_i|$ is the number of nodes in EC_i ($1 \leq i \leq q$), and then makes nodes in each EC_i have the same QI attribute values and degree. G_A preserves the privacy of G against an adversary's attack described in Sect. 3.2. However, when G is anonymized into G_A, information loss is necessarily incurred. Before we describe the detailed algorithm, we discuss the information loss in the next subsection.

3.4 Information Loss

The information loss consists of two factors: the loss of content information and the loss of structural information. The loss of content information incurs when the QI attribute values of nodes are changed, whereas the loss of structural information incurs when edges are changed. In other words, the information loss can be viewed as the difference between G and G_A. As the information loss increases, the utility of G_A decreases. Therefore, when anonymizing G into G_A, we should minimize the information loss in order to maximize the utility of G_A. In this section, we describe how to measure the information loss for given G and G_A.

Suppose that G is anonymized into G_A by partitioning nodes in G into $EC_1, \cdots, EC_q$ and making nodes in each EC_i ($1 \leq i \leq q$) have the same QI attribute values and degree. Because nodes in each EC_i are modified to have the same QI attribute values and degree, the loss of content and structural information incurs.

3.4.1 Loss of Content Information

To measure the loss of content information, we adopt the measurement introduced in [5]. The content information of a node consists of numerical attributes and hierarchical attributes. A numerical attribute contains a numeric value e.g., age and salary, whereas a hierarchical attribute contains hierarchical data e.g., ZIP code and sex. The hierarchy of hierarchical data can be represented by a taxonomy tree, as shown in Fig. 2. In a taxonomy tree, an ancestor node represents a more generalized concept than its descendent nodes and a descendant node represents a more specialized concept than its ancestor node. Let $N_1, \cdots, N_m$ be numerical attributes in QI. Let $C_1, \cdots, C_n$ be hierarchical attributes in QI. For each equivalence class EC_i, the proposed method makes nodes in EC_i have the same value for $N_1, \cdots, N_m$, and $C_1, \cdots, C_n$, respectively. For each N_j ($j = 1, \cdots, m$), the proposed method anonymizes the value of N_j of each node in EC_i to a range $[\mathrm{MIN}_{Nj}, \mathrm{MAX}_{Nj}]$, where MIN_{Nj} and MAX_{Nj} are the minimum and maximum values of N_j of nodes in EC_i, respectively. For each C_k ($k = 1, \cdots, n$), the proposed method anonymizes the value of C_k of each node in EC_i to $LCA(C_k, EC_i)$, where $LCA(C_k, EC_i)$ is the least common ancestor of the values of C_k of nodes in EC_i.

Because the proposed method modifies the QI attribute values of nodes, the loss of content information incurs. For an equivalence class EC_i, let $CL(EC_i)$ be the loss of content information caused by making nodes in EC_i have the same QI attribute values. Then, $CL(EC_i)$ is calculated as follows:

$$CL(EC_i) = |EC_i| \cdot \sum_{j=1}^{m} \frac{(MAX_{N_j} - MIN_{N_j})}{dom(N_j)} + \sum_{k=1}^{n} \frac{N_{L_N}(S_{C_k}(LCA(C_k, EC_i)))}{N_{L_N}(T_{C_k})}, \quad (1)$$

where $dom(N_j)$ is the size of the domain of a numerical attribute N_j, T_{C_k} is the taxonomy tree of a hierarchical attribute C_k, $S_{C_k}(r)$ is the sub-tree of the taxonomy tree of C_k whose root is r, and N_{L_N} is the number of leaf nodes of a tree T. Now we define the total loss of content information.

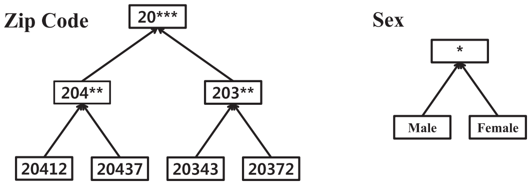


Fig. 2 Examples of taxonomy trees.

Definition 2. (**TLC: Total Loss of Content information**) Let G be a social network and G_A be an anonymized social network of G, which is produced by partitioning nodes of G into equivalence classes $EC_1, \cdots, EC_q$ and making nodes in each EC_i ($i = 1, \cdots, q$) have the same QI attribute values. Then, the total loss of content information caused by anonymizing G into G_A, denoted by $TLC(G, G_A)$, is defined as follows: (Here, $\mid QI \mid$ is the number of attributes in QI.)

$$TLC(G, G_A) = \frac{1}{\mid QI \mid} \sum_{i=1}^{q} CL(EC_i) \tag{2}$$

□

Recall that the information loss consists of the loss of content information and the loss of structural information. Because the loss of content information increases as the number of attributes in QI increases, we normalize it by dividing by $\mid QI \mid$ in order to prevent the loss of structural information from being relatively disregarded.

3.4.2 The Loss of Structural Information

For each equivalence class EC_i, the proposed method makes nodes in EC_i have the same degree. To do this, our method adds extra edges to nodes in EC_i to make all of them have the same degree $D_{MAX}(EC_i)$, where $D_{MAX}(EC_i)$ is the maximum degree of nodes in EC_i. The detailed procedure to add extra edges to nodes in EC_i to make them have the same degree will be described in the next section. Because the proposed method adds extra edges to nodes in EC_i, the loss of structural information incurs. We measure the loss of structural information based on the number of edges added to G. Let $SL(EC_i)$ be the loss of structural information caused by making nodes in EC_i have the same degree. Then, $SL(EC_i)$ is calculated as follows: (Here, $D_{MAX}(EC_i)$ is the maximum degree of node in EC_i and D_j is the degree of the j^{th} node in EC_i.)

$$SL(EC_i) = \sum_{j=1}^{|EC_i|} D_{MAX}(EC_i) - D_j \tag{3}$$

Because the proposed method makes nodes in EC_i have the same degree $D_{MAX}(EC_i)$, $D_{MAX}(EC_i) - D_j$ represents the number of edges added to the j^{th} node in EC_i. Now we define the total loss of structural information as follows:

Definition 3. (**TLS: Total Loss of Structural information**) Let G be a social network and G_A be an anonymized social network of G. Then, the total loss of structural information caused by anonymizing G into G_A, denoted by $TLS(G, G_A)$, is defined as follows:

$$TLS(G, G_A) = \sum_{i=1}^{q} SL(EC_i) \tag{4}$$

3.4.3 Total Loss of Information

Definition 4. (**TL: Total Loss of information**) Let G be a social network. Let G_A be an anonymized social network of G, which is produced by partitioning nodes in G into equivalence classes $EC_1, \cdots, EC_q$ and making nodes in each EC_i ($i = 1, \cdots, q$) have the same QI attribute values and degree. Then, the total loss of information caused by anonymizing G into G_A, denoted by $TL(G, G_A)$, is defined as follows: (Here, r is the user-defined weight parameter.[†])

$$TL(G, G_A) = r \cdot TLS(G, G_A) + (1 - r) \cdot TLC(G, G_A) \tag{5}$$

4. The Anonymization Algorithm

Our algorithm produces an anonymized social network G_A, where each node has at least (k - 1) other nodes indistinguishable from it, while considering the information loss discussed in Sect. 3.4. Our anonymization algorithm consists of two steps: '*EC Grouping and Value Generalization*' and '*Graph Construction*'.

4.1 EC Grouping and Value Generalizaion

Before we generalize the values, we partition the nodes in G into ECs in such a way to minimize TL. Because the problem of partitioning a set of elements into disjoint subsets and minimizing a given objective function is NP-hard [8], we heuristically partition the nodes in G into ECs as follows. First, we randomly pick a node n_1 from G and make an EC with n_1. Next, we pick a node n_2 ($\neq n_1$) from the remaining nodes of G that minimizes TL when added to EC, and add n_2 to the EC. We iterate this node-picking-process until the EC has k nodes. When the EC has k nodes, we make another EC from the remaining nodes in G. We iterate this EC generation process until there are less than k nodes in G that are not contained in any EC. Since it is impossible to make a new EC from less than k nodes, we add each of the remaining nodes to the EC that minimizes TL if the node is added to the EC. After all nodes in G are partitioned into ECs, we generalize the node values in the same EC such that they have the same attribute value and degree value. The detailed algorithm is shown in Fig. 3.

Example 2. Let G be a social network in Fig. 1 and nodes in G are partitioned into $EC_1 = \{n_1, n_5\}$, $EC_2 = \{n_2, n_6\}$, and $EC_3 = \{n_3, n_4\}$. Then after EC Grouping and Value Generalization, G is changed into G_G as shown in Fig. 4.

4.2 Graph Construction

The algorithm of 'EC Grouping and Value Generalization' transforms the original graph G into G_G which satisfies the k-anonymity with regard to the content value and degree. Now, we need to construct a graph G_A, where the changed

---

[†] The user-defined weight parameter r ($0 \leq r \leq 1$) controls a relative importance between $TLS(G, G_A)$ and $TLC(G, G_A)$. This parameter enables users to decide the weight of the importance of structural information.

```
Algorithm EC Grouping and Value Generalization
  Input : an original graph G, k as in k-anonymity
  Output : C = { EC_1, EC_2, ... , EC_j }
  C = ϕ , i = 0 ( i ≤ j )
  repeat
      i ← i + 1
      n_r ← select a node randomly from G
      G ← G - {n_r}
      make an EC_i with n_r
      C ← C ∪ EC_i
      repeat
          n_m ← select a node that minimizes TL if n_m is contained to EC_i
          G ← G - {n_m}
      until |EC_i| = k
  until |G| < k                    // |G| denotes the number of nodes in G
  for each remaining node n in G
      insert n to EC that incurs minimum TL
  end-for
  for each EC_i in C
      for each n ∈ EC_i
          generalize n to have same attribute and degree values in the same EC_i
      end-for
  end-for
End-Algorithm
```

Fig. 3 Algorithm of EC Grouping and Value Generalization.

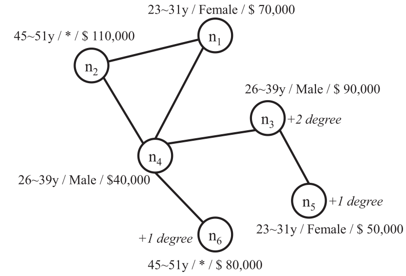


Fig. 4 A social network G_G converted from G.

degree values are reflected. (Note that the changed degree '+2' of node n_3 is not applied in Fig. 4.)

A naive approach to construct G_A is to add edges to the nodes whose degree values are changed. (i.e., n_3, n_5 and n_6 in Fig. 4.) However, this approach is not deterministic. For example, in Fig. 4, edges should be added to n_3, n_5 and n_6. Straightforwardly, we may try to add an edge between n_3 and n_5, and n_3 and n_6, respectively. However, because there already exists an edge between n_3 and n_5, we cannot make G_A by adding edges in such a way.

To deal with this problem, we may have to add additional edges between nodes whose degree values are changed and those whose degree values are not changed in G_G. However, it makes the problem more complicated. For example, adding an edge between n_1 and n_6 to make the degree value of n_6 equal to 2 affects the degree value of n_1, so we need to further consider the degree value of n_1 and those nodes in the *EC* to which n_1 belongs. Moreover, it is difficult to estimate that how many additional edges are needed to construct G_A. Hence, instead of adding edges to G_G, we propose a method called *SER* (*Selective Edge Removal*) that removes edges from a complete graph that has the same nodes as G_G. A complete graph is a graph in which every pair of distinct nodes is connected by an unique edge [1].

```
Algorithm SER
  Input : G_C = {N_C, E_C}, G_G = {N_G, E_G}
  Output : anonymized graph G_A
  // edge (n_1, n_2) means that candidate edge between n_1 and n_2
 1: for each e_c ∈ E_C
 2:       if (e_c ∉ E_G)
 3:             e_c is called candidate
 4:       end-if
 5: end-for
 6: repeat
 7:       n_c ← select the first node in NodeList
 8:       if ( D_current (n_c, G_C) > D_goal (n_c, G_G))
 9:             Edge Removal (n_c)
10:       end-if
11:       edges connected to n_c is removed from candidate group
12:       NodeList ← NodeList - {n_c}
13: until there are no more candidate edges left in G_G
14: G_A ← G_G
End-Algorithm

 Function Edge Removal (n_c)
15:   i ← D_current (n_c, G_C) - D_goal (n_c, G_G)
16:   j ← the number of candidate edges between n_c and a node contained in the same EC as n_c
17:       if ( i > j )
18:             p ← i
19:                   for (s = 0 ; s < j ; s++)
20:                         if ( p ≠ 0 && there is no more candidate edge connected to n_c )
21:                               Change degree value (n_c, p)
22:                               goto line 47
23:                         else
24:                               E_C ← E_C - {edge (n_c, a node contained in the same EC as n_c)}
25:                               p ← p - 1
26:                         end-if
27:                   end-for
28:                   for (t = 0 ; t < i - j ; t++)
29:                         if ( p ≠ 0 && there is no more candidate edge connected to n_c )
30:                               Change degree value (n_c, p)
31:                               goto line 47
32:                         else
33:                               E_C ← E_C - {edge (n_c, a random node in EC )}
34:                               p ← p - 1
35:                         end-if
36:                   end-for
37:       else
38:             for (v = 0; v < i ; v++)
39:                   if ( p ≠ 0 && there is no more candidate edge connected to n_c )
40:                         Change degree value (n_c, p)
41:                         goto line 47
42:                   else
43:                         E_C ← E_C - {edge (n_c, a node contained in the same EC as n_c)}
44:                         p ← p - 1
45:                   end-if
46:             end-for
47:       end-if
 End-Function

 Function Change degree value (n_c, p)
48:  for each node n_s in the same EC as n_c
49:        D_current (n_s, G_G) ← D_goal + p
50:  end-for
51:  Rearrange NodeList
 End-Function
```

Fig. 5 Algorithm of SER.

SER starts with a complete graph G_C whose nodes are the same as G_G. Then *SER* selectively removes edges that are not necessary to construct G_A. Although deletion of an edge also affects to the degree of nodes connected to the edge, finding unnecessary edges in a finite edge set is simpler than adding edges to graph G_G. The detailed algorithm is as follows:

For the complete graph G_C of G_G where G_C and G_G have the same nodes, we classify the edges of G_C into two types - (1) the original edges that also exist in G_G, (2) new edges that do not exist in G_G. Since we should preserve the edges of G_G in G_A, we selectively remove the edges of the latter type. We call the edges of the latter type in G_C as *candidate* edges.

In the algorithm (Fig. 5), we maintain a list *NodeList* of nodes in G_G, where nodes are ordered in the descending order of their changed degree in G_G. For example, *NodeList* = [n_3, n_4, n_1, n_2, n_5, n_6] in Fig. 4. Also, we use the notations $D_{current}(n_i,\ G_C)$ and $D_{goal}(n_i,\ G_G)$, where $D_{current}(n_i,\ G_C)$

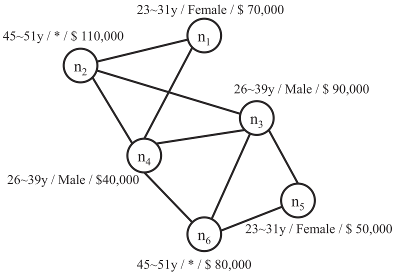


Fig. 6 The anonymized social network G_A from G.

is the current degree of n_i in G_C and $D_{goal}(n_i,\ G_G)$ is the changed degree value of n_i in G_G.

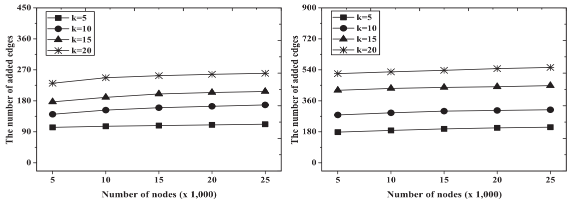
Next, we pick the first node n_C in the list *NodeList*. Then we remove *candidate* edges connecting n_C and a node contained in the same *EC* as n_C until $D_{current}(n_i,\ G_C)$ equals to $D_{goal}(n_i,\ G_G)$. However, there is a case where there are no more *candidate* edges connecting n_C and a node contained in the same *EC* as n_C, but $D_{current}(n_i,\ G_C)$ is larger than $D_{goal}(n_i,\ G_G)$. In this case, if n_C has *candidate* edges connected to it, we randomly remove *candidate* edge connected to it until $D_{current}(n_i,\ G_C)$ equals to $D_{goal}(n_i,\ G_G)$. However, if n_C has no more *candidate* edges connected to it, we should stop removing edge from n_C. In this case, for each node n_S of n_C and the nodes contained in the same *EC* as n_C, we change $D_{goal}(n_i,\ G_G)$ to $D_{current}(n_i,\ G_C)$, since $D_{current}(n_i,\ G_C)$ is the minimum degree value for n_C to construct G_A. When $D_{current}(n_i,\ G_C)$ equals to $D_{goal}(n_i,\ G_G)$, the edges connected to n_C are removed from the *candidate* group. Then we remove n_C from *NodeList*, and pick the next node in *NodeList*. We iterate the same processes until there are no more *candidate* edges left in G_C.

After the SER process, we get the anonymized network G_A, which preserves the nodes and edges of the original graph. Figure 6 shows the anonymized social network G_A from G.

5. Performance Experiments

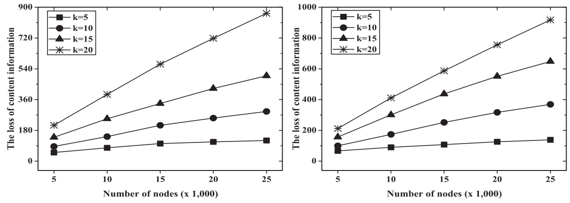
5.1 Datasets

For our experiments, we use the 'Adult' dataset from the UC Irvine Machine Learning Repository [2]. We consider 5 *QI* attributes; 'Age', 'Sex', 'Race', 'Education' and 'Native country'. As described in the Sect. 3.4.1, *QI* attributes are classified into numerical *QI* attributes and hierarchical *QI* attributes. The 'Age' attribute is a numerical *QI* attribute and the 'Sex', 'Race', 'Education' and 'Native Country' attributes are hierarchical *QI* attributes. Hierarchical attributes have their own pre-defined taxonomy trees. Since the dataset contains only the content information, we generate edges such that a social network has the property of a 'scale-free graph' [4]. A scale-free graph is a graph whose degree distribution follows the 'power' law. To measure the scalability of our proposed method, we perform experiments with various numbers of nodes (5000, 10000, 15000, 20000 and 25000). In order to investigate the effects of the density of social networks, we perform experiments with a variety of numbers of average degree per node (3, 7, 10 and 20). We also perform experiments with different k values (5, 10, 15 and 20), which stand for the level of privacy.



(a) $AD = 10$ (b) $AD = 20$

Fig. 7 The number of added edges in G_A by our method.



(a) $AD = 10$ (b) $AD = 20$

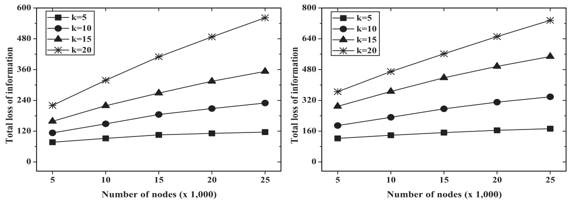
Fig. 8 The loss of content information in by our method.

5.2 Experiment Results

To anonymize the structural information, our method adds extra edges. Roughly speaking, the number of added edges represents the amount of change of the structural information. Figure 7 shows the number of edges added by our method. It is observed that the total number of added edges is increased as the number of nodes of the original social network is increased. However, the increase rate of the number of added edges decreases as the number of nodes increases. It means that our proposed method is scalable, so it can be applied to a large social network. A large k value leads to the increase of the number of added edges because more edges are needed to make all nodes in the *EC* have the same degree. *AD* (*Average Degree*) indicates the density of social networks. A large *AD* means that one person in the social network has many relationships with other persons. Since the distribution of nodes' degree becomes diverse with a large *AD*, the number of added edges increases as *AD* increases.

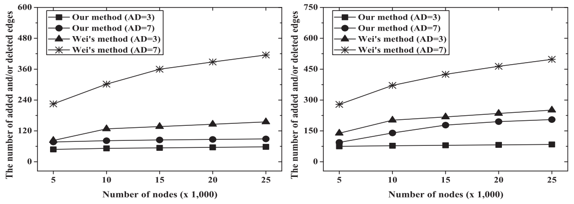
Figure 8 shows the loss of content information by the proposed method. It is observed that the loss of content information increases sub-linearly with the number of nodes. It means that our proposed method is also scalable in terms of the number of nodes.

Figure 9 shows the total loss of information by varying the number of nodes, k and *AD*. The total loss of information is represented as a sum of the total loss of structural information and the total loss of content information (We fixed $r = 0.5$, mentioned in $Definition$ 4). In Fig. 9, the total loss of information increases with the number of nodes.



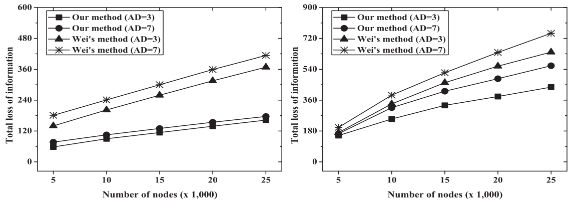
(a) $AD = 10$ (b) $AD = 20$

Fig. 9 Total loss of information by our method.



(a) $k = 10$ (b) $k = 20$

Fig. 10 Comparison of the amount of structural changes.



(a) $k = 10$ (b) $k = 20$

Fig. 11 Comparison of total loss of information.

However, with small k, there is almost no increase of the total loss of information, since it is easy to make *EC*s with more similar nodes. For the same reason, the total loss of information is sub-linearly increased with the increase of the number of nodes.

Figure 10 and Fig. 11 show the experiment results of comparing our proposed method with Wei et al.'s method [13]. The other methods, for example [6] and [15], are not suitable for our information loss metric. [6] clusters nodes and edges so it is not reasonable that measure their information loss by our metric. [15] focused on the content information of edges so that it cannot be measured by our information loss metric. Because our method considers the whole nodes of the social network when constructing *EC*s, whereas Wei's method considers node and its connected neighbor nodes for comprising k sub-graphs, our method reduces the total loss of information by grouping similar nodes together for anonymization.

6. Conclusion

In this paper, we addressed the problem of preserving privacy in a social network when an adversary knows both the content and structural information of the target victim. Existing methods for anonymizing a social network do not consider this type of privacy attacks, or distort the structural information of a social network too much during the anonymization process. We proposed a novel method for anonymizing a social network that prevents privacy attacks that use both the content and structural information. The proposed method prevents an adversary from disclosing the privacy of the target victim with a probability of more than $1/k$. Through extensive experiments we showed that the proposed method protects the privacy of social network data against content/degree attack, and outperforms the existing methods in terms of the amount of the information loss. In the future, we will extend our work to consider (1) an adversary who has more structural information, such as a subgraph of the social network, and (2) a dynamic social network in which insertions/deletions of nodes/edges are frequent.

References

[1] G. Agnarsson and R. Greenlaw, Graph Theory: Modeling, Application, and Algorithms, Pearson Education, 2007.

[2] A. Asuncion and D.J. Newman, UCIMachine Learning Repository, University of California, School of Information and Computer Science, Retrieved from the World Wide Web (http://www.ics.uci.edu/˜mlearn/MLRepository.html).

[3] L. Backstrom, C. Dwork, and J. Kleinberg, "Wherefore art thou r3579x?: Anonymized social networks, hidden patterns, and structural steganography," Proc. 16th international conference on World Wide Web, pp.181–190, 2007.

[4] A.L. Barabasi, Linked: The new science of networks, Cambridge, Perseus Publishing, 2000.

[5] J.W. Byun, A. Kamra, E. Bertino, and N. Li, "Efficient k-anonymization using clustering techniques," Proc. Database Systems for Advanced Applications, pp.188–200, 2007.

[6] A. Campan and T.M. Truta, "Data and structural k-anonymity in social networks," LNCS, 5456, pp.33–54, Springer, Heidelberg, 2009.

[7] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis, "Resisting structural re-identification in anonymized social networks," Proc. VLDB Endowment, vol.1, no.1, pp.102–114, 2008.

[8] D.S. Hochba, "Approximation algorithms for NP-hard problems," ACM SIGACT News, vol.28, no.2, pp.40–52, 1997.

[9] K. Liu and E. Terzi, "Towards identity anonymization on graphs," Proc. 2008 ACM SIGMOD Conference, pp.93–106, 2008.

[10] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "ℓ-diversity: Privacy beyond k-anonymity," ACM Trans. Knowledge Discovery from Data, vol.1, no.1, Article 3, 2007.

[11] L. Sweeney, "k-anonymity: A model for protecting privacy," International Journal on Uncertainty, Fuzziness and Knowledge-based System, vol.10, no.5, pp.557–570, 2002.

[12] B. Thuraisingham, "Privacy-preserving data mining: Development and directions," J. Database Management, vol.16, no.1, pp.75–87, 2005.

[13] Q. Wei and Y. Lu, "Preservation of privacy in publishing social network data," Proc. International Symposium on Electronic Commerce and Security, pp.421–425, 2008.

[14] X. Xiao and Y. Tao, "m-invariance: Towards privacy preserving republication of dynamic datasets," Proc. 2007 ACM SIGMOD Conference, pp.689–700, 2007.

[15] E. Zheleva and L. Getoor, "Preserving the privacy of sensitive relationships in graph data," LNCS, 4890, pp.153–171, Springer, Heidelberg, 2008.

[16] B. Zhou, Y. Han, J. Pei, B. Jiang, Y. Tao, and Y. Jia, "Continuous privacy preserving publishing of data streams," Proc. International Conference on Extending Database Technology, pp.648–659, 2009.

[17] B. Zou and J. Pei, "Preserving privacy in social networks against

neighborhood attacks," Proc. 2008 IEEE 24th International Conference on Data Engineering, pp.506–515, 2008.

[18] L. Zou, L. Chen, and M.T. Ozsu, "k-automorphism: A general framework for privacy preserving network publication," Proc. VLDB Endowment, vol.2, no.1, pp.946–957, 2010.

Min Kyoung Sung received the B.S. degree in Computer Science from Korea University, Seoul, Korea, in 2009, and he is currently in the integrated M.S. and Ph.D. course of Computer Science at Korea University. His research interests include privacy and graph databases

Ki Yong Lee received the B.S. and the M.S. degrees in Computer Science from KAIST, Daejeon, Korea, in 1998 and 2000, respectively, and his Ph.D. degree in Computer Science from KAIST in 2006. He was a senior engineer at Samsung Electronics, Korea, from 2006 to 2007. He was a research assistant professor of the Department of Computer Science at KAIST, from 2008 to 2009. He joined the faculty of the Department of Computer Science at Sookmyung Women's university, Seoul, Korea, in 2009, where currently he is an assistant professor. His research interests include database systems, data warehousing, OLAP, and embedded software

Jun-Bum Shin received the Ph.D. in Computer Science from KAIST, Daejeon, Korea, in 2003, and he is currently in Samsung Electronics. His research interests include security, cryptography, and privacy.

Yon Dohn Chung received the B.S. degree in Computer Science from Korea University, Seoul, Korea, in 1994, and the M.S. and Ph.D. degrees in Computer Science from KAIST, Daejon, Korea, in 1996 and 2000, respectively. He was an assistant professor in the Department of Computer Engineering at Dongguk University, Seoul, Korea, from 2003 to 2006. He joined the faculty of the Department of Computer Science and Engineering at Korea University, Seoul, Korea, in 2006, where currently he is an associate professor. His research interests include broadcast databases, XML databases, graph databases, distributed/parallel processing of large-scale data and privacy.