TRANSACTIONS ON DATA PRIVACY 11 (2018) 1 —22

Alpha Anonymization in Social Networks using the Lossy-Join Approach

Kiran Baktha*, B K Tripathy**
* Department of Electronics and Communication Engineering, VIT University, Vellore, Tamil Nadu
** Department of Computer Science and Engineering, VIT University, Vellore, Tamil Nadu
E-mail: sundarambaktha@hotmail.com, tripathybk@vit.ac.in

Abstract. Social networks contain important information about the society. Publishing them tends to have various advantages to data analyzers. However, privacy preservation in social networks has always been of primary concern. If these networks are published raw or with an ineffective anonymization technique, an adversary, even with limited background knowledge has the potential to extract the sensitive information present in the network. The Lossy-Join approach was used by Wong et al (2007) to develop an algorithm in order to achieve (α, k) - anonymity [14] in relational data. We extend this approach and develop an algorithm by using the same concept so that (α, k) - anonymity can be achieved in social networks. First, we run the proposed algorithm on a small network for illustration of its effect. Further, we test our approach on a real dataset from the United State Power grid and another large synthetic input in Erdős–Rényi graph generated by using R. Through experimental analysis, we establish that the efficiency of our algorithm is better than a general (α, k) - anonymity algorithm developed by Chakraborty et al [6] in 2016. We also propose a technique to add noisy sensitive labels into the model in case an anonymizer wishes a higher level of anonymization. The noise nodes required for anonymization are added so that they have minimal social importance.

Keywords. Social Networks, k-anonymity, l-diversity, alpha-anonymization, data anonymization, noise nodes, lossy-join, sensitive attributes

1 Introduction

Social networks are rapidly growing day by day. Networks such as Facebook, Instagram and LinkedIn generate a huge amount of data containing vital information about user behavior, spread of diseases and so on. But protection of

privacy in these networks is of foremost importance to prevent user identification and leakage of private information. A structure attack occurs when the structural information, most importantly the degree and subgraph of a node are used to link it and access sensitive information based on available background knowledge. To prevent these attacks, the *k*-anonymity model was proposed in [13]. A network satisfies *k*-degree anonymity if for every node there are at least *k*-1 other nodes in the network with the same degree as the node. Let us consider a small raw graph provided in Fig.1 that is needed to be anonymized. The transformed graph in Fig. 2 has 2-degree anonymity. An edge was needed to be added between the nodes 4 and 5 to ensure that nodes 4 and 8 have the same degree 3. However, this model had a significant drawback when sensitive labels are taken into account. For, if a group of nodes with the same degree have the same sensitive label then it becomes easy for an adversary to obtain the sensitive label. For example, in Fig. 2 both the nodes with degree of 3 (nodes 4 and 8) have the same sensitive label. To avoid this effect *l*-diversity was proposed in [10]. A network satisfies *l*-diversity if for every equivalence group, there exist at least *l* distinct sensitive labels in the group. Fig. 3 shows a raw graph with $k = 2$ and $l = 2$. We see that both the nodes of degree 3 have the same sensitive level "Heart Attack" in Fig.2. So, we add an edge between nodes 4 and 6 so that now we have two nodes of degree 3 (nodes 6 and 8) with different levels ("Heart Attack" and "Aids). Node 4 now has degree 4 and in its group we have two other nodes with different levels.

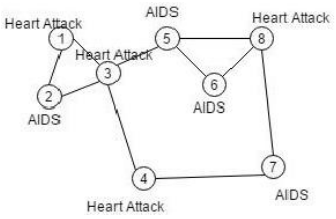

Figure 1. Raw Graph

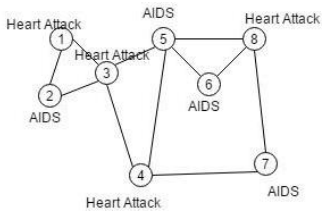

Figure 2. 2-degree anonymity

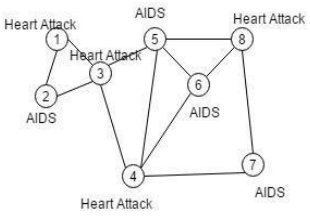

Figure 3. 2-degree- 2-diversity anonymity

The majority of the techniques used to anonymize social networks are edge-editing ([7, 9]), clustering ([3, 18]) and noise node addition ([5, 6, 16]). Clustering approach involves merging a subgraph to one super node but this method causes the node-label relations to be lost. Edge-editing approach involves the addition, deletion or swapping of edges but they sometimes modify the structural properties of a graph by connecting two far away nodes together or

by the deletion of a bridge link between two communities. Since in the noise node addition approach, noise nodes are added to achieve anonymization, the structural property of the original graph remains intact and hence the anonymized network has better utility. So, we use this approach to develop our anonymization method.

The k-anonymity model was proposed in order to prevent the re-identification of individuals in the released data set. However, it does not consider the inference relationship from the quasi-identifier (QID) to some sensitive attribute. A QID is a set of attributes that may serve as an identifier in the data set. For any QID Q, an equivalence class set, called a QID-EC is the collection of all the tuples in a table with identical values of Q. If it happens that all tuples in a QID-EC contain the same sensitive value in the released data set then any tuple in the QID-EC can be identified to have the particular sensitive value. This was termed as homogeneity attack. The (c, l) - diversity model, introduced by Machanavajjhala et al [10] in 2006 handles this attack. However, there is another attack called the "background knowledge attack", which was introduced in [10]. Here, the attacker is supposed to have some additional knowledge about the respondent, which helps him to identify the respondent from its QID-EC. The parameter l describes the level of diversity of sensitive values. The whole idea of using the two parameters c and l was to ensure that the most frequent sensitive value in a group should not be too frequent after the next p most frequent sensitive values are removed. But, it was observed by Wong et al [14] in 2006 that the setting of the parameters c and l by a user is quite difficult. Moreover, it is not very clear about the background knowledge an attacker can have. To keep background knowledge under control one must be prepared to eliminate large amount of possible values [14]. Moreover, it was noted by the authors that the algorithm in [10] is based upon a global-recoding exhaustive algorithm named Incognito, which is not scalable and may lead to higher distortion. To handle the above issues in a better way, the (α, k) - anonymity model was introduced by Wong et al [15] in 2007. This model requires that the value of the frequency (in fraction) of any sensitive value in any QID-EC is not more than α after anonymisation. This is in addition to k-anonymity and hence is named as (α, k) - anonymity. In [6], alpha anonymity was obtained by adding nodes to an equivalence group until the equivalence group satisfies alpha anonymity. Since the target degree of an equivalence group is chosen to be the degree of the highest degree node present in the group, the degrees of a lot of nodes are required to be changed in the group. This leads to high change in the cumulative degree of a group. In this paper, we

try to reduce the number of nodes in a group for which the degrees are to be changed by using a technique called the Lossy-Join. This concept was introduced in [15] to achieve (α, k) - anonymity in relational databases. In this approach, the basic idea is that if two tables with a join attribute are published, the join of the two tables can be lossy and this lossy join helps to conceal the private information. The authors split a table consisting of sensitive relational data into two tables and joining them generates several spurious tuples, which make the re-identification of the sensitive original table impossible. We are utilizing the same concept in this paper, where a sensitive network is split into another non-sensitive network (i.e. a network not having any sensitive attribute) and a table. The same lossy join approach helps us to achieve the desired anonymization. In our approach, we obtain (α, k) - anonymity by first making the graph satisfy k anonymity and then use lossy join technique to achieve alpha anonymization. This reduces the constraint in forming equivalence groups, thereby reducing the change in the cumulative degree change required for an equivalence group to achieve the target degree for all the nodes in the group. This actually reduces the number of noise nodes that are needed to be added to the network. But, our approach leads to the loss of node-label relationship and this is a tradeoff we had to do so that the structural property of the graph is least disturbed and also to reduce the number of noise nodes to be added. We use the eigenvector centrality concept over the traditional degree centrality concept as a measure when grouping nodes in equivalence groups for k-anonymity. This prevents the mixing of highly influential nodes with lower influential nodes in the same equivalence group. We further introduce the concept of a noisy sensitive label that can be added to provide α-anonymization.

Our approach in this paper comprises of two steps:

Before using these two steps, the nodes are arranged in order of their eigenvector values to preserve the social importance of nodes, which prevents the existence of an equivalence group with both high and low influential nodes.

- **Step-1:** Eigenvector centrality concept is used to manage k-anonymity and then the sensitive labels are replaced in the network with sensitive class numbers.
- **Step-2:** The lossy join approach is used to achieve alpha anonymity

We first test our approach on a simple network comprising of 8 nodes and 10 edges. Then the algorithm is applied on a synthetic dataset of higher complexity. In order to test the behaviour of the process of noise node addition a real dataset is used. The results obtained are compared with those of the alpha

anonymization algorithm for social networks proposed in [6]. The idea of comparing with the results of only [6] instead of any other algorithm proposed in this direction is that besides being the only algorithm providing alpha anonymity for social networks, it was experimentally established in [6] that it is better than the other existing anonymisation models. In fact, our aim is to compare our results with those of [6] and show that our proposed algorithm is superior to it.

The rest of the paper is organized as follows: Section 2 states the existing techniques developed for anonymization in social networks. Section 3 defines key terminologies. Section 4 introduces our proposed approach in detail. Section 5 contains information of the datasets and Section 6 discusses on the experimental results obtained by applying our algorithm and that in [6] on the different data sets as stated earlier. Also, a comparative analysis of these results is provided to show the superiority of our proposed algorithm over that in [6].

2 Literature Review

Clustering [3, 8] and edge-editing [7, 9] techniques have been the two major approaches used in social network anonymization. Preservation of the structural property of social networks should be given importance during anonymization. In [8] a heuristic clustering model was proposed which uses sub-graph and vertex refinement methods to prevent the disclosure of sensitive attribute values. Here, the *k*-degree anonymity model for social networks was proposed. In [4], a *p*-sensitive *k*-anonymous clustering model was proposed. In [18], a *k*-neighborhood model was proposed which requires that for each node in the graph, there should be at least *k*-1 isomorphic neighborhoods. The *l*-diversity model was proposed in [19] to handle the drawbacks associated with *k*- anonymity. Also, in this paper two other models; namely the (c, *l*)-diversity model and the entropy *l* -diversity model were introduced. In [12], Shrivastava et al. proposed an algorithm where the noise nodes could be detected by computing the triangle probability difference between the noise nodes and the normal nodes. In [17], Zheleva et al. studied the possibility of an attacker exploiting the published information of the actors to access the sensitive or unpublished information while mining social network data. In [11], Narayanan et al. presented a brief analysis on the possibility of actor identification by an attacker in the anonymized graph, if the attacker has background knowledge about another graph which overlaps partially with the anonymized network. In

[5], an introductory notion to anonymize social networks has been discussed. In [16], Yuan et al. proposed a noise node addition model which preserves the structural properties of a graph. In [6], Chakraborty et al. proposed (α, l) and recursive (α, c, l) diversity techniques which uses eigenvector centrality concept as well as the noise node addition concept defined in [16] to create an anonymized network. Most of the concepts stated in this section are presented in section 3.

3 Key Terminologies and Problem Description

The following are certain key terminologies used in the paper:

i) Social Network: A social network can be considered as a graph G consisting of a 4 tuple (V, E, σ, λ), where

V: The set of vertices in the graph and each vertex represents a node in the network,

E: The set of edges between the vertices and E $\subseteq$ V × V,

σ: denotes the set of sensitive labels associated with the vertices V and

$\lambda: V \rightarrow \sigma$ denotes the mapping of the vertices to their sensitive labels.

For example, in Fig. 1, V = {1, 2…., 8}, σ = {Heart Attack, AIDS}, $\lambda(1) = \lambda(3) = \lambda(4) = \lambda(8) = \text{"Heart Attack"}$ and $\lambda(2) = \lambda(5) = \lambda(6) = \lambda(7) = \text{"AIDS"}$

ii) Equivalence Group: Equivalence group is a group of nodes that have the same degree. In Fig. 1 nodes 1, 2, 4, 6 and 7 form an equivalence group because all of them have the same degree of 2.

iii) Eigenvector Centrality: Eigenvector centrality was first introduced in [1] (Also, see [2]). Consider a network S which has an adjacency matrix A of size $n \times n$. Then the eigenvector centrality e_i of the i^{th} node is the i^{th} entry of the normalized eigenvector corresponding to the largest eigenvalue of the network. Let λ be the largest eigenvalue and $x = (x_1, x_2, ..., x_n)$ be the corresponding eigenvector. Then by definition, we have $Ax = \lambda x$ and hence $x = \left(\frac{1}{\lambda}\right) Ax$. So, we get

$$x_i = \left(\frac{1}{\lambda}\right) \sum_{j=1}^{n} a_{ij} x_j, \quad i = 1, 2, ..., n$$

The main advantage of eigenvector centrality of a node is that it depends not only on the degree of a node but also on the position of the node and the

influence of the neighboring nodes of that node in the network. Therefore, eigenvector centrality concept can be used to measure the importance of a node in the entire network.

iv) Eigenvector Centrality based Sequence (EVCS): Instead of arranging the nodes by their order of occurrence, we use the eigenvector centrality of each node to arrange them. The eigenvector centrality sequence of a network consists of four tuples (id, evc, d, s), where id is the node identity, evc is the eigenvector centrality, d is the degree and s is the sensitive label associated with that node. The EVCS of the nodes (n[1], n[2], …, n[m]) is the decreasing order representation of the nodes based on the eigenvector centrality of the nodes i.e. (n[1].evc ≥ n[2].evc ≥ ….n[m].evc). Suppose the graph G has the node sequence as (1, 0.6, 2, Heart Attack), (2, 0.8, 3, AIDS) and (3, 0.4, 1, AIDS) then the EVCS of G is (2, 0.8, 3, AIDS), (1, 0.6, 2, Heart Attack) and (3, 0.4, 1, AIDS).

The EVCs for the toy graph in Figure 1 are: Node 1 -> 0.6025261, Node 2 -> 0.6025261, Node 3 -> 1.0000000, Node 4 -> 0.5621129, Node 5 -> 0.8925138, Node 6 -> 0.6192658, Node 7 -> 0.4950400 and Node 8 -> 0.7545345.

v) α -deassociation property: The α-deassociation property was originally introduced in [14]. In [6] the authors have extended the property to social networks. Consider a social network graph G having an equivalence group E and (E, s) be the set of vertices in E having the sensitive label s. Then the equivalence group satisfies α-deassociation property for a particular s if the relative frequency of 's' in that equivalence group is less than α, i.e. $\lceil |(E, s)| / |E| \rceil \leq \alpha$, where α is a user specified threshold value lying between 0 and 1. A graph is said to satisfy the α-deassociation property if all the equivalence groups satisfy the α-deassociation property.

vi) (α, k) and (α, l) anonymity: A social network graph is said to be (α, k) anonymized if for every equivalence group in the network there exists at least k- 1 other nodes of the same degree and the α-deassociation property is satisfied. A social network satisfies (α, l) anonymity if every equivalence group has l distinct sensitive attributes after satisfying (α, k) anonymity.

vii) Noise Nodes and Noisy Sensitive labels: A noise node is a node not originally present in the network but is added to the network for anonymization purposes. Similarly, a noisy sensitive label is a sensitive label different from the sensitive labels in the network but it is the one that belongs to the same generalization that is added to the network for anonymization purposes.

ix) Non-sensitive graph (NSS graph): A non-sensitive graph is a social network graph with all the sensitive labels removed and replaced by a sensitive

class.

x) Sensitive table (SS table) and Sensitive Class: A sensitive class contains nodes of the same degree in a network. Sensitive table consists of the class id and the respective sensitive labels in the class.

4 Anonymized Graph Construction

In this section, we describe our proposed approach in constructing an anonymized graph.

4.1 *k* - Anonymous degree sequence generation

First, we generate a degree sequence using all the nodes of the social network graph that is *k*-anonymous. *K-anonymous* property needs to be satisfied before we can apply the Lossy-Join approach because an adversary with some background information on the degree information about the node might be able to identify the sensitive label. For example, if there is only one node with degree 4 in the network and the adversary knows a person with degree 4 then he can relate him/her to the person in the network and easily predict the sensitive label. To prevent this, we apply the *k*-anonymisation first. Then we use the eigenvector centrality concept to arrange the nodes in the order of their importance. In the algorithm, G_{temp} is a temporary group and G_{ki} is the i^{th} equivalence group that satisfies *k*-anonymity. The algorithm is presented below:

Algorithm: k-degree sequence generation
Input: EVCS
Output: k-anonymous degree sequence

1. Set i = 0, G_{temp} = {}
2. **while** |EVCS| > 0 **do**
3. **while** (| G_{temp} | < k && |EVCS| > 0) **do**
4. G_{temp} = G_{temp} ∪ EVCS [0]
5. Remove EVCS [0] from EVCS
6. end **while**
7. **if** (| G_{temp} | ≥ k) **then**
8. Set G_{ki} = G_{temp} && increment i by 1
9. Delete the elements in G_{temp}
10. **else**
11. $G_{k(i-1)}$ = $G_{k(i-1)}$ ∪ G_{temp}
12. Delete the elements in G_{temp}
13. end **if-else**
14. end **while**

We first create a temporary group G_{temp} and add nodes into it until the size of the group reaches k and we have more nodes to add from EVCS. Once the group reaches a size k, we move it to the actual output group at index i which is zero initially, increment i and remove all the nodes in the temporary group. Steps 11 and 12 handle the case when the temporary group has less than k nodes and we have no more nodes to add. In this case, we merge it with the previously created group. For the toy example shown in Fig. 1, the EVCS would be {3, 5, 8, 6, 1, 2, 4, 7} meaning that Node 3 has the largest eigenvector centrality value and Node 7 has the least. The General Approach in [6] would create two groups [3, 5, 8, 6] and [1, 2, 4, 7] with target degrees 4 and 2 respectively which leads to the following modifications required: node 5 - increase degree by 1, node 6 - increase degree by 2 and node 8 - increase degree by 1. Our Approach would create two groups [3, 5, 8], [6, 1, 2, 4, 7] with targets 4 and 2 respectively which leads to the following modifications required: node 5 - increase degree by 1, node 8 - increase degree by 1. We can observe that our approach requires fewer disturbances than the General Approach in [6].

4.2 SS table generation

After generating the *k*-anonymized graph, we remove the sensitive labels in the graph and replace it with a class id of i for the i^{th} equivalence group. We further introduce the concept of noisy sensitive label that can be added to the sensitive class in-case we run out of sensitive labels to be added to satisfy α-deassociation property. Another added advantage is that the anonymizer need not keep track of the graph's sensitive labels to choose a lower limit of α. Any value of alpha between 0 and 1 can be used in this approach irrespective of the network. The noisy sensitive labels are selected such that these are similar to the existing sensitive labels in the network.

For example, if the graph is published by a hospital and the sensitive labels are all diseases then the noisy sensitive labels can be other diseases not present originally in the network i.e. they belong to the same generalization. The procedure involved is described below:

Algorithm: SS Table Generation
Input: k-anonymous degree sequence
Output: SS Table

1. Let S_{net} = {distinct sensitive labels of the whole network}.
2. Let S_i = {distinct sensitive labels of the i^{th} equivalence group}.
3. for every equivalence group *i*

For every equivalence group, S_{temp} will contain the sensitive labels not present in the group but in the network. Once we run out of sensitive labels to add and we need more to satisfy α-deassociation property, we add noisy sensitive labels. The main use of SS Table is to reduce α-deassociation on the graph which leads to lower number of noise nodes and lesser disturbance to the raw graph. The SS Table for the toy example in Fig. 1 is described in Table. 1.

4.3 Graph Construction

In this section we discuss various graph construction techniques like neighbourhood edge editing technique, adding noise nodes to decrease degree, adding noise nodes to increase degree and adjusting noise node degree.

4.3.1 Neighbourhood edge editing technique

In this step, we first try to change degrees of nodes by adding or deleting edges such that the distance between the two nodes changes by 1 only. This algorithm works in 3 cases:

- **Case 1:** If the degree of a node u is needed to be increased and the degree of another node v is needed to be decreased to attain the target degree such that u and v are direct neighbors then we randomly select a direct neighbor w of v that is not connected to u, add an edge (u, w) and remove an edge between w and v.
- **Case 2:** If there are two nodes u, v which are two hop neighbors and both need to increase their degree to attain the target degree then we add an edge between u and v if there was no edge initially
- **Case 3:** If there are two nodes u and v such that their degrees are required to be decreased to attain the target degree and removing an edge still makes these two nodes two hop neighbors, then this edge is removed.

As far as the toy example is considered, in both the General Approach and our approach this algorithm does not modify the network because none of the above cases are applicable.

4.3.2 Adding noise nodes to decrease degree

If the degree of a node u is needed to be decreased by n to attain its target degree, then a noise node is created and connected to u. Then (n+1) edges connecting node u with its neighbors are deleted and the noise node is connected to all those neighboring nodes. For example, suppose it is required to

decrease the degree of node u by 'one'. This procedure is carried out as shown in Fig. 4. Here, n_1 and n_2 are the 2 neighbours of u whose link to u be deleted and are connected to u through the noise node N.

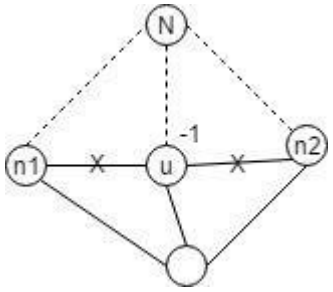

Figure 4. Adding noise node to decrease degree

In the toy example, for both General Approach and our approach this algorithm is not invoked as for no node its degree is needed to be decreased.

4.3.3 Adding noise nodes to increase degree

If there is a node u that needs to increase its degree to attain its target degree, then create a noise node and connect it to u. If there is another node within two hops distance from u that also needs to increase its degree, then we connect that node to the noise node. This process is repeated until u achieves its target degree or the noise node reaches a target degree. If the latter happens, we create another noise node and repeat the procedure. Suppose the degrees of two nodes u and v are required to be increased. Then we create a new noise node N and connect these nodes to the new noise node as illustrated in Fig. 5.

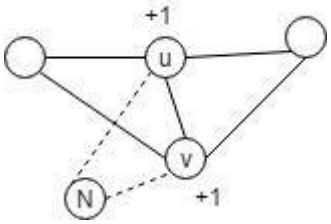

Figure 5. Adding noise node to increase degree

Using General Approach for the toy example, this algorithm creates a noise node N and adds nodes 5, 8 and 6 to it because all are two hop neighbors. Now,

node 6 is disconnected as the degree of N has become 2 after joining to nodes 5 and 8 which is the target degree. After this, two more noise nodes are created and both are connected to node 6. Now, all the nodes reach their target degree. Totally three noise nodes are added.

In our approach, this algorithm creates a noise node N and adds nodes 5 and 8 to it. Now all the nodes reach their target degree with only one noise node being added.

4.3.4 Adjusting the noise node degree

If there are noise nodes whose degrees are not equal to that of any one of the target degrees among the equivalence groups in the network, we build a link for each pair of these nodes. We then select any one target degree that is even times greater than the degree of an even degree noise node or odd times greater than an odd degree noise node and assign it as the target degree of the noise node. In order to increase the noise node degree, we find a link (u, v) having minimum average distance from the noise node to the nodes u and v. We then remove the link between u and v and connect them through the noise node. This process is repeated for all the noise nodes, which are not having a suitable initial target degree. The noise nodes are then put under the equivalence group whose target degree is equal to their degrees. This procedure can be seen in Fig.6 where the link (u, v) is removed and the nodes u and v are connected to the noise node which increases its degree by 2. That is why we need to choose an odd target degree for an odd degree noise node and even target degree for an even degree noise node.

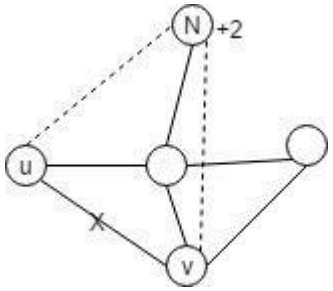

Figure 6. Adjusting noise node degree

Considering General Approach for the toy example, this algorithm connects the two noise nodes added to node 6 as shown in Fig. 8 so that they both have a degree of 2 which is one of the target degrees. The sensitive labels attached to

these noise nodes do not matter as any label seems to satisfy the alpha-deassociation property. For descriptive purposes, we assigned AIDS to all the noise nodes. In our approach, this algorithm does not modify the graph as there is only one noise node which already has the target degree of 2. We assign the sensitive class S2 to it because its target degree is 2 which is the target degree of the second equivalence group.

The work flow diagram of the anonymization procedure is given in Fig.7 below. We start by generating the EVCS sequence and then continue until we reach the end of noise node adjustment algorithm. The sections in the paper where the procedures are described are provided in brackets under each step.

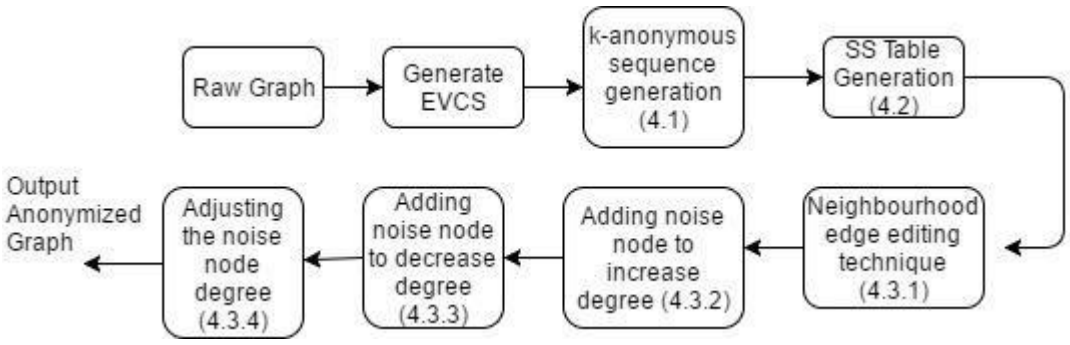

Figure 7. Flow chart of the anonymization procedure

5 Datasets used

The data used for analysis are

- **Small Network**: We have used a small social network graph in Fig. 1 as a toy example, which was also used in [6] to verify our approach.
- **Real Dataset**: We use the United States Power Grid dataset which contains information about the power grid of the western states in the United States of America. An edge represents a power line and a vertex is a generator, a transformer or a substation. This network comprises a total of 4,941 vertices and 6,594 edges.
- **Synthetic Dataset**: We have used the software R to generate the Erdős–Rényi model graph with 3,000 vertices and 6,000 edges. An Erdős–Rényi graph is a type of random classical graph where N number of vertices are connected by M number of edges from a total set of N(N-1)/2 edges. In this paper, we refer it as ER graph.

6 Results and Analysis

We use Average Path Length (APL) and the number of edge modifications required as metrics to test the performance of our approach. We implemented the proposed anonymization procedure by using the programing language Java.

Average Path Length (APL) – It is defined as the average distance between all pairs of nodes in the entire network. Mathematically, it is represented as:

$$APL = (2/(N(N-1))) \sum_{n_i, n_j \in G} d(n_i, n_j)$$

where N is the number of vertices in graph G and $d(n_i, n_j)$ denotes the distance between the pair of nodes n_i and n_j. APL can be used as a measure of the efficiency of information transportation in a network.

In Fig. 8 is the (α, l) anonymized network for the network in Fig. 1 obtained by using the approach mentioned in [6]. We have been mentioning this approach as the General Approach for indicative purposes. The value of α is taken as 0.6, k as 3 and l as 2.

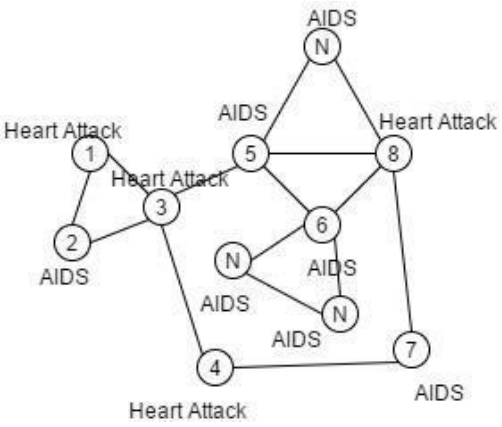

Figure 8. Alpha anonymization using General Approach

Fig. 9 is the graph obtained by using the Lossy Join approach. The SS table has been included along with the graph to achieve α-anonymization.

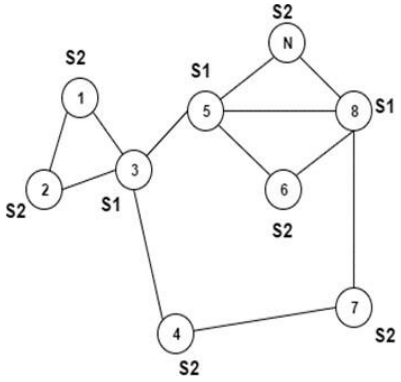

Figure 9. Anonymized graph using Lossy Join

Sensitive Class	Sensitive Labels
S_1	Heart-Attack, AIDS
S_2	Heart-Attack, AIDS

Table 1. SS Table

The number of noise nodes is reduced in Lossy Join approach as compared to the General approach. The (α, k) anonymization using Lossy Join approach also satisfies the (α, l) anonymization proposed in [6] because an equivalence sensitive class has at least k distinct sensitive labels and the value of l should always be below k. To illustrate the added benefit of noisy sensitive label let us consider the case where the anonymizer specifies the value of l to be 3 or α to be below 0.5. Then the general approach will not be able to anonymize the network. One might argue that we can use the noisy sensitive attribute in the General Approach and assign noisy sensitive label to the noise nodes. But in most of the cases there might be only one noise node with a degree say 3 and adding this one noise node to the equivalence group having a target degree of 3 might not be enough. Further, higher target degree equivalence groups in large networks generally do not have a noise node added in their group. In these cases, all the equivalence groups do not have l diverse labels or satisfy α-deassociation property and the network is not α-anonymized.

Suppose the raw graph in Fig.1 is a graph published by a hospital and the sensitive labels are all diseases, then the noisy sensitive table could be a list of labels shown in Table 2. Then the Lossy Join approach would generate Table 3 as the SS table if l is desired to be 3. The network is the same as in Fig.5 with no

additional requirement of any noise nodes.

Flu
Cancer
Malaria
Organ Failure
Diabetes

Table 2. Noisy Sensitive Table

S_1	Heart-Attack, AIDS, Flu
S_2	Heart-Attack, AIDS, Cancer

Table 3. SS Table for l = 3

We have also run our algorithm on the Power Grid and Erdős–Rényi models. The results are depicted in figures 10 to 15 below. The value of α has been taken as 1/2 for Power grid model and 1/3 for Erdős–Rényi model. l has been kept at a constant level of 3 for all the cases. The sensitive labels are generated from R software randomly and vary from A-Z. Hence, the networks have 26 sensitive labels in total.

US Power Grid dataset results:

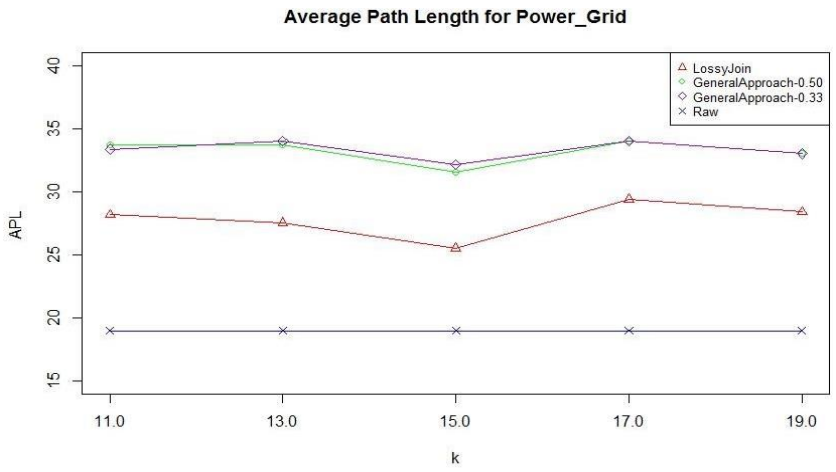

Figure 10. APL of the anonymized Power Grid graph

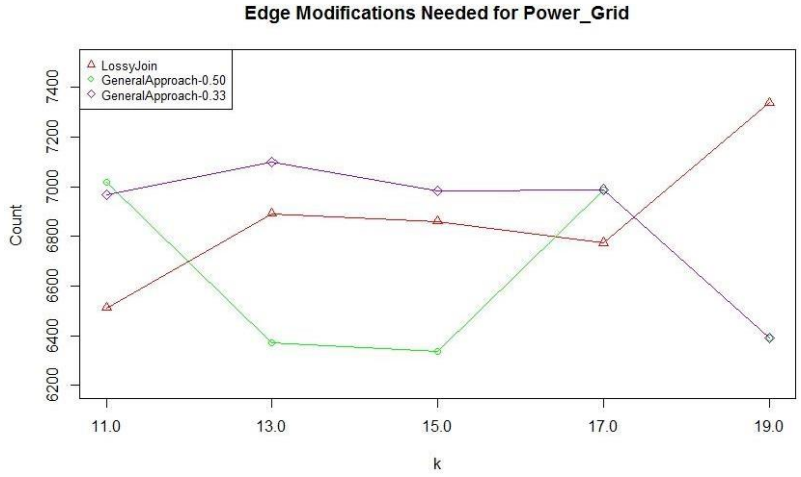

Figure 11. Number of Edge Modifications Required for Power Grid graph

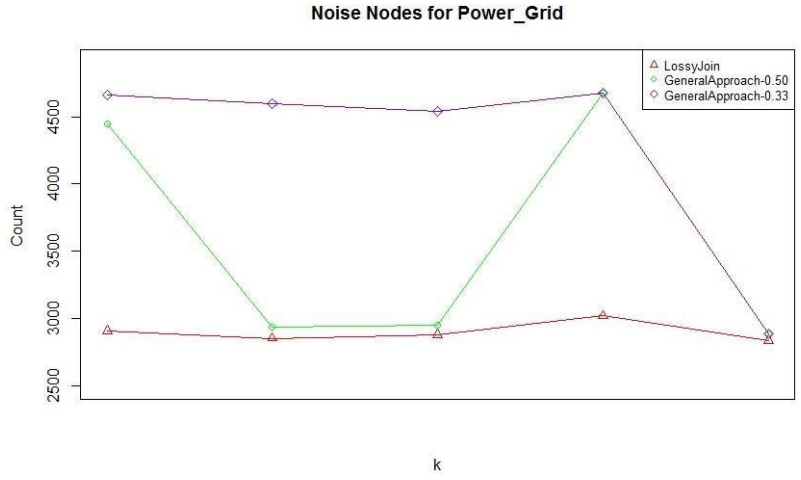

Figure 12. Number of Noise Nodes added for Power Grid graph

Erdős–Rényi dataset results:

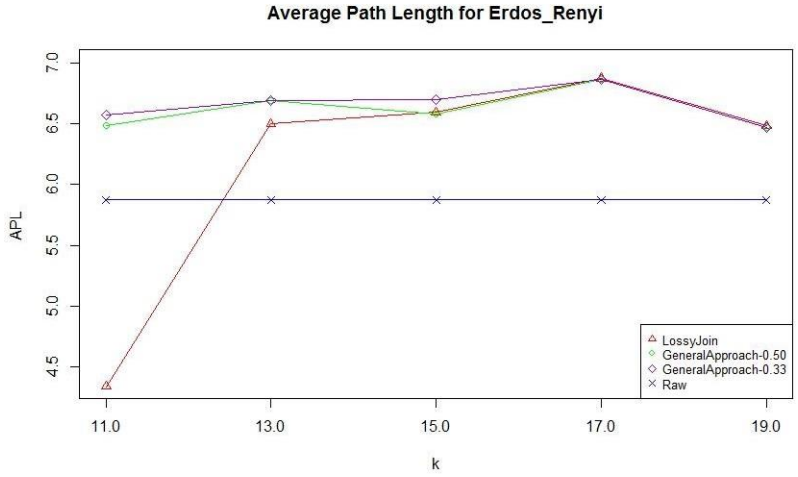

Figure 13. APL of the anonymized ER graph

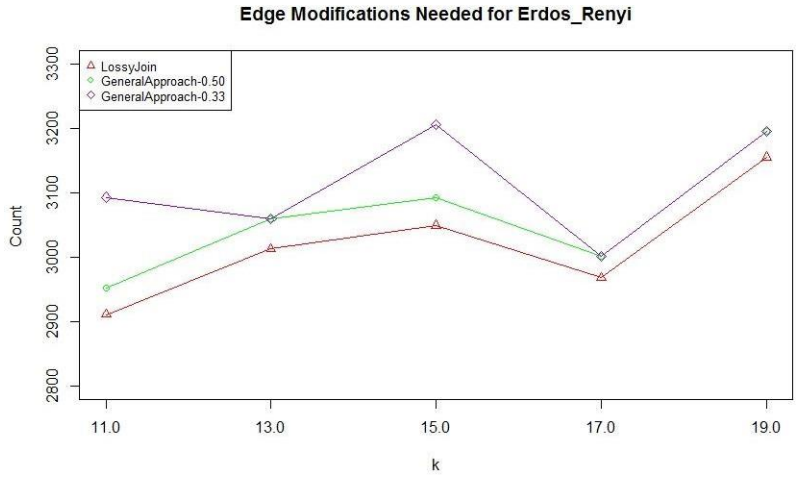

Figure 14. Number of edge modifications needed for ER graph

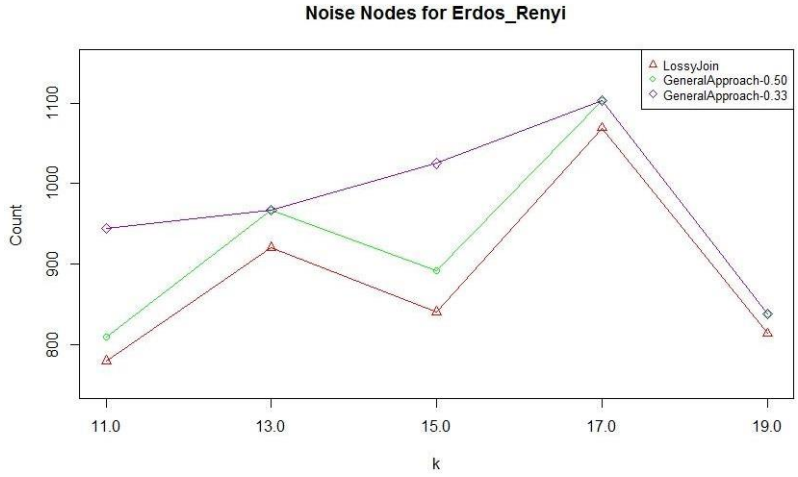

Figure 15. Number of Noise Nodes added for ER graph

The average path length in both Power Grid and ER graph (Fig. 10 and Fig. 13 respectively) indicate that Lossy-Join approach induces lesser disturbances to the raw graph as the APL values are nearer to those of the raw graphs in all the cases. The number of noise nodes added in both these datasets is lower as well for the Lossy-Join approach in all the cases (Fig. 12 and Fig. 15). Lossy-Join approach required higher number of edge modifications (Fig. 11) than the General approach in a few cases, with α equal to 0.5 for the Power Grid dataset. But, in these cases, the neighbourhood edge editing technique (4.3.1) has modified more internal edges, which leads to addition of lesser number of noise nodes. We have to keep in mind that in the Lossy-Join approach the output graph is same for any value of α and only the SS Table generated would vary. The number of edge modifications required remained lower in Lossy-Join than the General approach for all cases in the ER graph (Fig. 14).

7 Conclusion and Future Work

In this paper, we have used an alternative approach to the alpha anonymization method suggested in [6]. Our approach leads to lesser amount of noise nodes being added and also the number of edge modifications required is reduced in most of the cases. It also provides additional flexibility to the anonymizer through the added concept of noisy sensitive attribute. The noise node addition algorithms are followed as per [16] so that these noise nodes are added in an

intelligent manner to have minimum social importance. As a future scope, alpha anonymization can be studied further on weighted and directed graphs. Handling multiple sensitive labelled nodes can also be investigated on. Estimation of utility of data in the anonymized graph and the effect of noise node addition on such data can be studied further as well.

References

[1] Bonacich, P.: Factoring and weighting approaches to status score and clique identification, Journal of Mathematical Sociology, vol.2, (1972), pp. 113–120.

[2] Bonacich, P.: Some unique properties of eigenvector centrality, Social Networks, 29, (2007), pp. 555–564.

[3] Campan, A. and Truta, T.M.: A clustering approach for data and structural anonymity in social networks, Proceedings of the Second ACM IGKDD International Workshop on Privacy, Security and the First ACM SIGKDD Int'l Workshop on Privacy, Security and Trust in KDD, (PinKDD '07) (2007), pp.153-171.

[4] Campan, A., Truta, T.M. and Cooper, N.: P-Sensitive K-Anonymity with Generalization Constraints, Trans. Data Privacy, (2010), vol. 2, pp. 65-89.

[5] Chakraborty, S. and Tripathy, B.K.: Privacy Preservation in Social networks through alpha – anonymization techniques, Proc. of the 2015 IEEE/ACM Int'l Conf. on Advances in Social Networks Analysis and Mining (ASONAM 2015), (2015), pp 1063-1064

[6] Chakraborty, S. & Tripathy, B.K.: Alpha-anonymization techniques for privacy preservation in social networks, Social Network Analysis and Mining, (2016), 6: 29, pp1-11.

[7] Cheng, J., Fu, A.W.-c. and Liu, J.: K-Isomorphism: Privacy Preserving Network Publication against Structural Attacks, Proc. Int'l Conf. Management of Data, (2010), pp. 459-470.

[8] Hay, M., Miklau, G., Jensen, D., Towsley, D. and Weis, P.: Resisting Structural Re-Identification in Anonymized Social Networks, Proc. VLDB Endowment, vol. 1, (2008), pp. 102-114.

[9] Liu, K. and Terzi, E.: Towards Identity Anonymization on Graphs, SIGMOD'08: Proc. ACM SIGMOD Int'l Conf. Management of Data, (2008), pp. 93-106.

[10] Machanavajjhala, A., Kifer, D., Gehrke, J. and Venkitasubramaniam, M.: l-Diversity: Privacy beyond K-Anonymity, ACM Trans Knowledge Discovery Data, vol. 1, article 3, (2007).

[11] Narayanan, A. and Shmatikov, V.: De-Anonymizing Social Networks, Proc. IEEE 30th Symposium Security and Privacy, (2009), pp. 173-187

[12] Shrivastava, N., Majumdar, A. and Rastogi, R.: Mining (Social) Network Graphs to Detect Random Link Attacks, Proc. IEEE 24th Int'l Conf. Data Eng. (ICDE '08), (2008), pp.486-495.

[13] Sweeney, L.: K-Anonymity: A Model for Protecting Privacy, Int'l J. Uncertainty Fuzziness Knowledge-Based Systems, (2002), vol. 10, pp. 557-570.

[14] Wong, R. C., Li, J., Fu, A. W. and Wang, K.: (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing, Proc. ACM SIGKDD, (2006), pp. 754-759

[15] Wong, R. C.-W., Liu, Y., Yin, J. Huang, Z., Fu, A. W.-C. and Pei, J.: (alpha, k)-anonymity based privacy preservation by lossy join, in Proceedings of the Joint 9th Asia-Pacific Web and 8th International Conference on Web-Age Information Management Conference on Advances in Data and Web Management, Springer, Huang Shan, China, (2007), pp. 733–744.

[16] Yuan, M., Chen, L., Yu, P.S. and Yu, T.: Protecting Sensitive Labels in Social Network Data Anonymization, Knowledge and Data Engineering, IEEE Transactions on, (2013), vol.25, no.3, pp.633-647.

[17] Zheleva, E. and Getoor, L. : To Join or Not to Join: The Illusion of Privacy in Social Networks with Mixed Public and Private User Profiles, Proc. 18th Int'l Conf. World Wide Web (WWW '09), (2009), pp. 531-540.

[18] Zhou, B. and Pei, J.: Preserving Privacy in Social Networks Against Neighborhood Attacks, Proceedings of the IEEE 24th International Conference Data Eng. (ICDE '08), (2008), pp. 506-515.

[19] Zou, L., Chen, L. and Ozsu, M.T.: K-Automorphism: A General Framework for Privacy Preserving Network Publication, Proc. VLDB Endowment, vol. 2, no.1, (2009), pp. 946-957.