Privacy-preserving Data Publishing in the Cloud: A Multi-level Utility Controlled Approach

Balaji Palanisamy and Ling Liu[†]

School of Information Sciences, University of Pittsburgh, College of Computing, Georgia Tech[†]

bpalan@pitt.edu, lingliu@cc.gatech.edu

***Abstract*—Conventional private data publication schemes are targeted at publication of sensitive datasets with the objective of retaining as much utility as possible for statistical (aggregate) queries while ensuring the privacy of individuals' information. However, such an approach to data publishing is no longer applicable in shared multi-tenant cloud scenarios where users often have different levels of access to the same data. In this paper, we present a privacy-preserving data publishing framework for publishing large datasets with the goals of providing different levels of utility to the users based on their access privileges. We design and implement our proposed multi-level utility-controlled data anonymization schemes in the context of large association graphs considering three levels of user utility namely: (i) users having access to only the graph structure (ii) users having access to graph structure and aggregate query results and (iii) users having access to graph structure, aggregate query results as well as individual associations. Our experiments on real large association graphs show that the proposed techniques are effective, scalable and yield the required level of privacy and utility for user-specific utility and access privilege levels.**

I. Introduction

Publishing data that contains sensitive information about individuals is an important problem. Conventional data publication schemes are targeted at publishing sensitive data either after a k-anonymization process [9], [10] or through differential privacy constraints [24] to allow users to perform ad-hoc analysis on the data. However, such an approach to data publishing becomes no longer applicable in shared multi-tenant cloud scenarios where users often have different levels of access to the same data. In tabular datasets, each record has a number of attributes, some of which identify or can potentially identify individuals (e.g., social security number, address, Zip-code, Birth-date) and some of which are potentially sensitive (e.g., disease or salary). Private data also arises in the form of associations between entities such as the drugs purchased by patients in a pharmacy store. Here, the associations between the entities (such as the drugs purchased by a patient) are considered sensitive and they can be naturally represented as large, sparse bipartite graphs with nodes representing the drugs and the patients and the edges representing the purchases of the drugs made by the patients. Such data may be of high value and importance for a number of purpose. For instance, medical scientists may want to study the outbreaks of new diseases based on the types of drugs purchased by patients; drug manufacturers may wish to perform business analytics based on the sale trends and purchase patterns of various drugs.

To effectively limit disclosure, we need to measure the disclosure risk of the published information. The conventional focus on the problem of data anonymization has been on tabular data, via *k-anonymization* [9], [10] and subsequent variations [13], [23], [17]. While a number of variations of such techniques focused on perturbing the graph structure to minimize disclosure risks, some existing work had also concentrated on retaining the graph structure but preventing the inference of the connections between individuals (represented by nodes in the graph) [6], [3]. The idea behind these techniques is to group nodes of the graph into disjoint sets and to expose the associations only between the groups instead of individual nodes.

In this paper, we argue that most existing work on privacy-preserving data publication targets at releasing safe versions of the dataset to provide accurate results to aggregate queries while protecting individual associations. Such data releases implicitly assume that all users of the data share the same access privilege levels which is no longer applicable in shared multi-tenant cloud storage systems where users often have different levels of access to the same data. For instance, in a drug purchase association graph, one may need to protect privacy and utility at different protection levels depending on the access privilege of the users. While some users (e.g., less privileged data analysts) may be allowed to obtain only graph structure related queries, some others (e.g., medical scientists) may have access to results of aggregate queries in addition to graph structure. In the same way, some highly privileged users (like the pharmacy store manager) may have access to even the individual associations in the graph that is considered sensitive to be exposed to other users. Current data publication schemes release a version of the dataset that can provide privacy-utility tradeoffs for at most one of the above-mentioned privacy/utility levels. If an association graph is anonymized to provide aggregate query results while retaining exact graph structure, then a user who does not have access to the graph structure will still be able to obtain graph structure information leading to increased disclosure of private information and similarly, a highly privileged user who needs to access the individual associations in the graph will be unable to do so since the individual association information is lost during the anonymization process.

In this paper, we propose a novel framework for anonymizing large association graphs with the goals of supporting multi-level utility-privacy tradeoffs based on user access privilege rights. In contrast to existing anonymization techniques that lose information during the anonymization process, our proposed schemes retain the sensitive information in an anonymous form. Privileged users having access to higher level of sensitive information can obtain it through a proposed key-based data access mechanism. First, we develop a key-based reversible graph structure perturbation technique that prevents less privileged users from accessing graph structure while allowing the original graph structure to be restored by privileged users. We use a safe grouping technique for grouping nodes of the graph to support aggregate queries. Finally, we devise a key-based node label permutation mechanism that allows the original ordering of the nodes to be restored such that highly privileged users obtain individual associations in the graph. To the best of our knowledge, this is the first research effort that is aimed at developing a systematic approach to supporting multi-level utility controlled private queries on anonymized datasets. The rest of the paper is organized as follows: We describe the multi-level association graph anonymization problem and various existing approaches in Section II. Section III presents a suite of key-based reversible graph anonymization techniques that protect privacy at different utility and privacy levels. Section IV presents our experimental evaluation on real large association graphs. We discuss related work in Section V and we conclude in Section VI.

II. Background and Preliminaries

In this section, we review the fundamental concepts related to association graphs and define the multi-level graph anonymization problem. We also discuss various known approaches and their limitations to protecting sensitive information in multi-level access limited scenarios.

We assume a bipartite graph dataset represented by $G = (V, W, E)$. The graph G consists of $m = |V|$ nodes of first type and $n = |W|$ of second type and a set of edges $E \subseteq V \times W$. For instance, the bipartite graph could represent the associations of patients and drugs based on the purchases made by them. In that case, the set of nodes, V represents patients and W represents drugs and any edge (p, d) in E will represent the association that the patient p bought the drug d. In a similar way, the bipartite graph G can represent the papers written by authors. In that case, the set V would represent the authors and the set W would represent the papers and edges will represent which papers were co-written by a set of authors. Other examples of such association relationship include movies watched by viewers, courses taken by students, places visited by people etc. We show an example of such a bipartite graph in the figure in Table 1 where the nodes represent drugs and patients and the edges represent the drugs purchased by the patients. The details of the patients and drugs are shown in Tables I and II and the associations captured by the bipartite graph is shown in Table III. We assume there are users with four different levels of access, each having its own privacy-utility requirement. It leads us to classify the possible queries that can operate on the published graph.

TABLE I: Patients

PID	DOB	Sex	Zipcode
P1	7/18/87	F	30323
P2	2/17/83	M	30323
P3	5/07/77	M	30327
P4	1/5/76	F	30328
P5	8/4/82	M	30330
P6	3/9/79	M	30331
P7	4/10/64	M	30331
P8	2/6/81	F	30334
P9	7/14/72	F	30337
P10	9/25/74	M	30338
P11	4/28/80	M	30338
P12	3/12/78	M	30339

A. Multi-level Utility/ Privacy Model

In general, there are 4 types of queries that can operate in a large association graph dataset.

- Type 0 - Queries on graph structure: these queries require only the graph structure to be answered accurately. E.g.: the maximum number of drugs purchased by an individual customer.
- Type 1 - queries involving attribute predicates on one side. E.g.: the average sales on antibiotic drugs in Zipcode 95123?
- Type 2 - queries involving attribute predicates on both sides. E.g.: total number of antibiotic drugs bought by patients in the city of Pittsburgh.
- Type 3 - queries involving actual associations. E.g.: who are the buyers of the drug, 'Setraline' ?

This classification of the queries leads us to define the privacy levels of various users.

1) Privacy levels: We consider 4 levels of privacy/utility corresponding to the 4 types of queries described above.

- Level 0 - No access users: users having no access to the dataset including graph structure, queries involving predicates and actual associations.
- Level 1 - Graph Structure users: these users obtain accurate answers to type 0 queries but do not get approximate or accurate answers to type 1 and type 2 queries.
- Level 2 - Aggregate query users: level 2 users have access to queries of type 0, type 1 and type 2. However these users can not access individual associations and hence do not get results for type 3 queries.
- Level 3 - All access users: users obtain accurate answers to all 4 types of queries. It represents the highest level of access to the dataset.

Next we discuss some existing approaches for graph anonymization and their applicability for the multi-level graph access problem considered in this work.

B. Existing techniques

Existing techniques for sensitive graph data publication can be broadly classified into two categories namely (i) *k-anonymity*-based graph anonymization and (ii) *differential privacy*-based graph publication.

TABLE II: Drugs

CID	Drug name	Category
D1	epinephrine	bronchodilator
D2	ibuprofen	analgesic
D3	Zovirax	antiviral
D4	Tylenol	analgesic
D5	erythromycin	antibiotic
D6	cortisone	steroid
D7	gentamicin	antibiotic
D8	insulin	hypoglycemic
D9	sertraline	antidepressant
D10	tramadol	analgesic
D11	cetirizine	antihistamine
D12	zolpidem	hypnotic

TABLE III: Patients-Drugs Association

PID	CID
P1	D5
P2	D8
P2	D9
P5	D11
P7	D5
P9	D3
P9	D12
P11	D11

1) k-anonymity-based approaches: A large number of existing techniques anonymize data based on the concept of *k-anonymization* [8], [9]. A direct application of tabular anonymization to graph data would require the graph to be represented using three relations. For instance, we can create three tables (Table I - Table III) for the patient-drug association graph shown in the figure in Table 1. In the k-anonymization process, first all patient and drug information that serves as quasi-identifiers are removed and then the drugs bought by the patients are grouped so that there are k or more subjects within each k-anonymity group. The k-anonymized version of this table must use generalization and suppression to ensure that each row is indistinguishable from $k-1$ other subjects [8], [9]. A more sophisticated approach to graph anonymization is to group the nodes of the graph to create disjoint groups so as to hide the individual association between the nodes of different groups [5]. This technique preserves the underlying graph structure, but masks the exact mapping from entities to nodes, so for each node we know a set of possible entities that it corresponds to. Unfortunately, both the above-mentioned k-anonymity approaches are not suitable when all users do not share the same access privilege levels as these schemes lose the sensitive information during the anonymization process.

2) Differential privacy-based techniques: An alternate approach to *k-anonymity*-based data publication is to release statistics of the dataset through differential privacy constraints [21]. Precisely, the differential privacy constraint ensures that the published statistical data does not depend on the presence or absence of an individual record in the dataset. Recent work had focused on publishing graph datasets through differential privacy constraints so that the published graph maintains as much structural properties as possible as the original graph while providing the required privacy [16]. However, such statistical data publishing does not support multi-level utility control as considered in our problem setting. Therefore, when aggregates are released through differential privacy constraint, the released information can match the privacy/utility needs for at most one privacy/utility level and hence users at other privacy/utility levels either encounter increased disclosure of private information or obtain information at reduced utility levels. To overcome these limitations, we propose a suite of key-based multi-level anonymization schemes that retain sensitive information in the anonymized version so that privileged users de-anonymize it on the fly through a key-based access control mechanism.

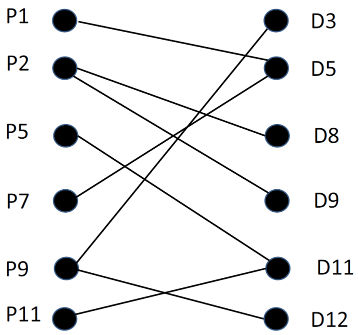


Fig. 1: Original graph

III. Multi-level Graph Anonymization

This section presents our proposed key-based multi-level anonymization schemes for supporting multi-level utility/ privacy tradeoffs in association graphs. We begin with an illustrative example for the association graph shown in Figure 1. The multi-level anonymization process applies a series of anonymization and perturbation techniques on the raw graph to obtain the final perturbed graph. The anonymized graph corresponding to the privacy/utility of level 0 users is shown in Figure 2. As we know, level 0 users have the lowest access utility levels and therefore can not derive any utility in terms of graph structure or aggregate results or exact associations. Level 1 users possess the structure key and use that to decode the exact graph structure and thus level 1 users obtain accurate results to queries on graph structure (Figure 2). Similarly, level 2 users possess the utility key in addition to structure key and therefore obtain accurate answers to aggregate queries in addition to queries on graphs structure (Figure 3). In the same way, level 3 users use their association key to obtain access to exact associations in the graph, thereby obtaining the highest utility out of the dataset. In the subsequent sections, we will discuss the detailed techniques for protecting graph structure from level 0 users, protecting aggregate query results from level 1 users and protecting individual associations from level 2 users respectively.

A. Protecting graph structure

The first step in the multi-level anonymization process is to obtain a reversible perturbation of the graph structure in order to protect the graph structure characteristics from level 0 users.

1) Protecting graph structure from level 0 users: The graph structure perturbation process begins with perturbing the associations between the nodes of the graph. It injects

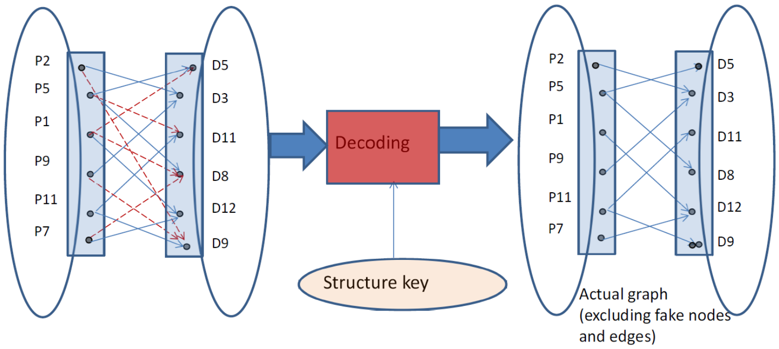


Fig. 2: Access for Level 1 users

a number of fake edges in the perturbed graph in order to hide the real structure of the underlying graph. The algorithm uses the graph structure key, K_s and another arbitrary integer to obtain a random seed which is used to generate a stream of pseudo-random numbers. This random stream drives the injection of fake edges into the graph. Concretely, if R_i is the i^{th} random number generated by a pseudo-random number generator, then the i^{th} fake edge in the graph is given by

$$(v, w) = (R_{2i} mod|V|, R_{2i+1} mod|W|)$$

We define R_i as the i^{th} non-colliding random number such that the random numbers R_i and R_{i+1} are able to form an edge $(R_{2i} mod|V|$,$R_{2i+1} mod|W|)$ such that $(R_{2i} mod|V|$, $R_{2i+1} mod|W|)$ is not a member of the original graph, G. But as we may note, the perturbation process may come across some R_i or R_{2i+1} such that $(R_{2i} mod|V|, R_{2i+1} mod|W|)$ already belongs to G. In such cases, the algorithm changes the seed of the pseudorandom generator to generate a different edge. Since the seed is generated as a function of the structure key and another arbitrary integer, m, the algorithm changes the integer, m in order to change the random seed. It therefore writes an entry (i, m) to a table T, where m denotes the integer used for generating the seed which is in turn used for generating the pseudorandom number, R_i. This table of entries is used during the de-perturbation process to restore the original graph when level 1 users provide the graph structure key, k_s.

The de-anonymization process works in a similar way. For decoding the perturbed graph, the level 1 users provide the graph structure key, K_s. The decoding process uses the same process for generating the random stream, R_i and uses this random stream to delete the fake edge $(R_{2i} mod|V|, R_{2i+1} mod|W|)$. From the seed table, T we know that, if there is an entry (i, m) in the seed change table, T, then the seed used for generating R_i should be derived using the integer, m and the structure key, k_s. Thus, the decoding process generates the exact stream of pseudo-random numbers and thus all the fake edges,$(R_{2i} mod|V|, R_{2i+1} mod|W|)$ are generated in the same sequence and deleted to restore the original graph.

Algorithm 1 Key-based Graph Structure Perturbation

```
$V, W$: Array of vertices
$E$: Set of edges in graph G
$K_s$: Graph structure key
$n$:Number of fake edges to add
$R_i$: $i^{th}$ random number
procedure GraphPerturb($G$, $K_s$, $n$)
    for $i = 1$ to $n$ do
        $m = 0$
        $seed = F(K_s, m)$
        $R_{2i} = PseudoRandom(2i, seed)$
        $R_{2i+1} = PseudoRandom(2i + 1, seed)$
        while $(R_{2i} mod|V|, R_{2i+1} mod|W|) \in E)$ do
            $seed = F(K_s, m)$
            $R_{2i} = PseudoRandom(2i, seed)$
            $R_{2i+1} = PseudoRandom(2i + 1, seed)$
            $m = m + 1$
        end while
        if $m > 0$ then
            add $T(i, m)$
        end if
    end for
end procedure
```

B. Protecting aggregate query results

The second step in the multi-level graph anonymization is to protect the aggregate query results from level 1 users. For example, level 1 users after decoding the graph structure may attempt to infer the results of aggregate queries if the graph maintains an ordering of the nodes based on some attributes. To overcome this, a global permutation is done on the nodes so that range queries involving attributes can not be inferred. We note that this global permutation needs to be done on both set of nodes V and W to prevent level 1 users from obtaining results to aggregate queries. The algorithm uses the utility key of level 2 users to generate a stream of pseudorandom numbers and this random stream drives the permutation of the nodes in the graph. Concretely, we use the modern version of FisherYates[6] shuffling algorithm to generate the permutation

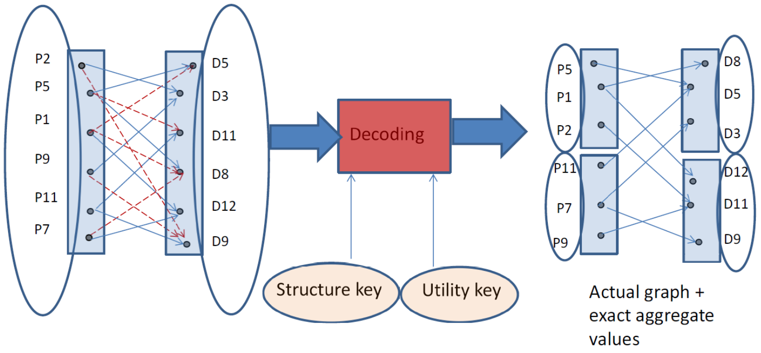


Fig. 3: Access for Level 2 users

Algorithm 2 Decoding Perturbed Graph

```
$V, W$: Array of vertices
$E$: Set of edges in graph G
$K_s$: Graph structure key
$n$:Number of fake edges to delete
$R_i$: $i^{th}$ random number
procedure GRAPHPERTURB($G$, $K_s$, $n$)
    for $i = 1$ to $n$ do
        if $T(i, m)! = null$ then
            $m = T(i, m)$
        else
            $m = 0$
        end if
        $seed = F(K_s, m)$
        $R_2 i = PseudoRandom(2i, seed)$
        $R_{2i+1} = PseudoRandom(2i + 1, seed)$
        remove edge $(R_{2i} mod|V|, R_{2i+1} mod|W|)$ from $E$)
    end for
end procedure
```

of the vertices. The algorithm generates a stream of pseudo random numbers using a seed generated by the aggregate utility key, K_u. For each i ranging from $|V|$ to 1, a pseudo random number, R_i is generated and the algorithm swaps the vertices $V[i], V[j]$. At the end of $|V|$ iterations, we obtain a random permutation of the ordering of the nodes. For decoding this random permutation, level 2 users use the aggregate utility key, k_u and generate the same seed that was used to drive the random permutation. When the seed is fed to the pseudo random number generator, it produces the same sequence of random numbers which indeed decides the set of vertices to be swapped in each iteration. Therefore, at the end of $|V|$ iterations, we obtain the original node ordering back. Note that randomly changing the ordering of the nodes prevents level 1 users from inferring aggregate query results. While level 2 users can decode this random ordering, we still need to ensure that level 2 users obtain only aggregate query results but not the individual associations in the graph. To enable this, the nodes in the graph are grouped into different sets of nodes prior to this global node label permutation process so that the associations between the nodes in the individual groups are safe to be exposed to level 2 users. We describe our proposed order-preserving safe grouping technique in the next subsection.

Algorithm 3 Key-based Attribute Permutation

```
$V$: Array of vertices
$K_u$: Aggregate utility key
$R_i$: $i^{th}$ random number
procedure NODESHUFFLE($V$, $K_u$)
    for $i = |V|$ down to 1 do
        $R_i = PseudoRandom(i, K_s)$
        $j = R_i mod|V|$
        $Swap(V[i], V[j])$
    end for
end procedure
```

Algorithm 4 Reverse Permuatation

```
$V$: Array of vertices
$K_u$: Aggregate utility key
$R_i$: $i^{th}$ random number
procedure NODESHUFFLE($V$, $K_u$)
    for $i = 1$ to $|V|$ do
        $R_i = PseudoRandom(i, K_s)$
        $j = R_i mod|V|$
        $Swap(V[i], V[j])$
    end for
end procedure
```

C. Protecting individual associations

The idea behind the proposed individual association protection mechanism is to group the nodes of the graph into disjoint sets and to expose the edges only between the different groups while the labels of the nodes in the individual groups are permuted to prevent the inference of the exact associations. Therefore, for a level 2 user, the exposed groups will enable to compute aggregate query results while the individual edges can not be inferred as the node labels within each group are permuted. In general such a grouping is called a k-grouping if the number of nodes in each group is greater than or equal to k [5]. Formally, a k-grouping of a graph is defined as follows:

Definition 1: Given a set V , a k-grouping is a function H mapping nodes to group identifiers (integers) so for any $v \in V$, the subset $V_v = \{v_i \in V : H(v_i) = H(v)\}$ has $|V_v| \geq k$. Formally, $\forall v \in V : \exists V_v \subseteq V : |V_v| \geq k \cup (\forall v_i \in V_v : H(vi) = H(v))$

That is, H partitions of V into subsets of size at least k. Thus the nodes are partitioned into sets of non-overlapping groups. Inside each group, the node labels are permuted using the same technique discussed in Section III-B so as to provide *k-anonymity*.

IV. Experimental Evaluation

Our experimental evaluation consists of evaluating both privacy as well as performance efficiency of the proposed anonymization schemes. We first evaluate the effectiveness of the proposed techniques in terms of the privacy protection by measuring the level of perturbation offered by them. We then evaluate the performance of the proposed key-based anonymization schemes in terms of anonymization and de-anonymization time for various privacy and utility levels.

The proposed anonymization and de-anonymization schemes are implemented as a Java library. The primary dataset used in the experiments is the DBLP data representing all conference publications. It is retrieved from *http://dblp.uni-trier.de/xml/*. The DBLP data set consists of $|V| = 402023$ distinct authors, $|W| = 543065$ distinct papers, and $|E| = 1401349$ (author, paper) edges. We consider all four types of queries and we use the following queries for each query type:

- Type 0 Query: Cumulative distribution of the number of papers of each author.
- Type 1 Queries: We use two type 1 queries: Query A: find the total number of authors in the publications satisfying predicate P_w, Query B: the total number of publications having only one author and satisfying predicate P_w.
- Type 2 Query: find the number of publications satisfying predicate P_w having authors satisfying P'_w.
- Type 3 Query: Is author x co-author of publication y?

We study the privacy and performance of the queries for various access privilege levels by varying a number of other parameters such as group size and the degree of graph structure perturbation.

A. Effect of Graph Structure Perturbation

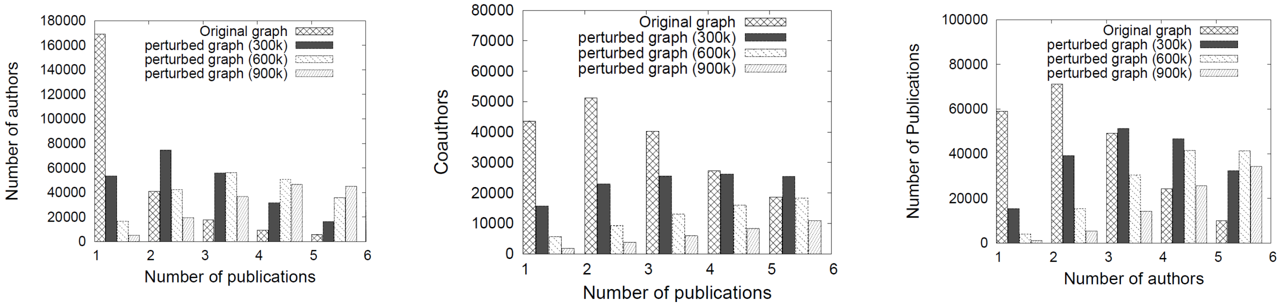
Our first set of experiments study the protection for graph structure provided by the key-based graph perturbation techniques on the utility for level 0 users. We measure three distributions related to the structure of the graph namely (i) the distribution of the number of authors based on the number of publications they have, (ii) the distribution of the number of co-authors based on the number of publications they have co-authored, (iii) the distribution of the number of publications based on the number of authors in them. Figure 4(a) represents the distribution of the authors based on the number of publications they have. The Y-axis represents the number of authors who have total publications shown in X-axis. The distribution is shown for different number of fake edges injected into the graph (ranging from 300,000 to 900,000). We find that when the graph structure is perturbed with randomly injected edges, the distribution is significantly altered. Thus level 0 users are not able to obtain the exact distribution present in the original graph. Also, we note that the distribution in the perturbed graph changes for different number of fake edges added into the graph. Therefore, if a randomly chosen number of fake edges are injected into the graph, an adversary does not have a clue on the amount of random perturbation done to the original graph and hence can not obtain accurate results to queries involving only graph structure. Here, the number of fake edges to be injected can be decided through a random distribution which is chosen based on the degree of perturbation required for the original graph. Similarly, Figure 4(b) shows the distribution of the number of co-authors (Y-axis) based on the number of publications they have co-authored (X- axis). Here also, we notice that the distribution is significantly changed after the graph perturbation process. We present the distribution of the number of publications based on the number of authors in the publications in Figure 4(c). The perturbed graphs again show that level 0 users do not obtain accurate results to this distribution.

B. Performance of Perturbation and Grouping techniques

Our next set of experiments is focused on studying the performance of the proposed key-based multi-level graph perturbation and grouping techniques based on anonymization time. We first study the time taken by the graph perturbation algorithm for various number of fake edges injected in the perturbed graph. Figure 5(a) shows the time taken by the graph structure perturbation process. The X-axis represents the percentage of fake edges injected compared to the total number real edges in the graph. We find that the perturbation process is quite fast with the average perturbation and de-perturbation time well within 10 seconds for the whole dataset. We present the time taken for the node grouping process in Figure 5(b) where the X-axis represents the group size, k. Here the value of l is kept as constant at 20. We find that the node grouping algorithm takes only a small amount time in the overall anonymization process. Similarly, the time taken for the node label permutation operation is shown in Figure 5(c) and it indicates that the process is quite fast and scales well for various group sizes.

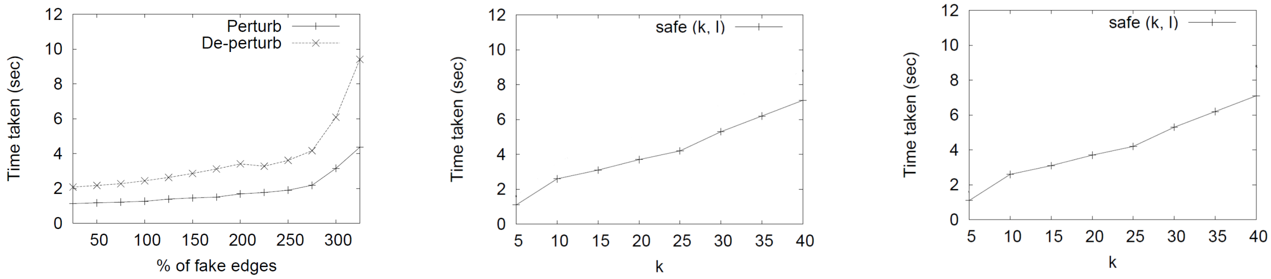
V. Related Work

The problem of information disclosure has been studied extensively in the framework of statistical databases. Samarati and Sweeney [8], [9] introduced the k-anonymity approach which has led to new techniques and definitions such as l-diversity [11], (α, k)-anonymity [20],t-closeness [15] and anonymization via permutation [22], [24]. However, these schemes are primarily targeted for data publishing with the goal to provide aggregate queries while protecting individual



(a) Distribution of No of papers (b) Distribution of collaborator count (c) Distribution of Publication count

Fig. 4: Effect of Graph Structure perturbation



(a) Edge Perturbation (b) Node grouping (c) Node label permutation

Fig. 5: Performance of perturbation and grouping techniques

information. There had been some work on anonymizing graph datasets with the goal of publishing statistical information without revealing information of individual records. In [25], the authors propose a graph anonymization scheme that ensures that each node has k others with the same (one-step) neighborhood characteristics to prevent unwanted disclosure. Ghinita et al. present an anonymization scheme for anonymizing sparse high-dimensional data using permutation based methods [10] by considering that sensitive attributes are rare and at most one sensitive attribute is present in each group. While most of the above mentioned work address the privacy risks in releasing unlabeled graphs, the safe grouping techniques proposed in [5], [2] consider the scenario of retaining graph structure but aim at protecting privacy when labeled graphs are released. However, as discussed earlier, these safe grouping techniques can not provide accurate results to aggregate queries. Also, the above-mentioned techniques are targeted at publishing a single safe version of the graph dataset which protects privacy at just one privacy/utility level.

Another promising direction of privacy research is represented by differential privacy techniques. Based on the concept of differential privacy introduced in [21], there had been many work focused on publishing aggregates of sensitive datasets through differential privacy constraints [23], [3], [7]. Differential privacy had also been applied to protecting sensitive information in graph datasets such that the released information does not reveal the presence of a sensitive element [12], [13], [18]. Recent work had focused on publishing graph datasets through differential privacy constraints so that the published graph maintains as much structural properties as possible as the original graph while providing the required privacy [16]. But, as mentioned earlier, these existing schemes do not support multi-level access to the same published dataset as the published dataset represents just one privacy level. To the best of our knowledge, our work presented in this paper is the first significant effort on providing multi-level privacy and utility control in a shared published dataset. We believe that many principles and ideas developed in this work will be complementary to both differential privacy-based as well as anonymity-based sensitive data publication schemes.

VI. Conclusion

This paper presents an anonymization framework for publishing large association graph datasets with the goal of supporting multi-level access controlled query processing in shared storage systems. Conventional data publication schemes target at releasing sensitive datasets through an anonymization process to support aggregate queries while protecting the information contained in individual records. We argue that such schemes are not suitable when different users have different levels of access to the same data. We propose a suite of anonymization techniques and a utility-preserving grouping technique to support multi-level access controlled query processing on published datasets. Our experiments on real association graphs show that the proposed techniques are efficient and scalable and support multi-level privacy-utility tradeoffs. Our future work is focused on applying the principles and concepts presented in this work to develop a multi-level private data publication scheme with differential privacy guarantees.

References

[1] C. Aggarwal. On k-Anonymity and the Curse of Dimensionality. In *VLDB*, 2005.

[2] S. Bhagat, G. Cormode, B. Krishnamurthy, D. Srivastava. Class-based graph anonymization for social network data. In *VLDB*, 2009.

[3] R. Chen Publishing Set-Valued Data via Differential Privacy. In *VLDB*, 2011.

[4] G. Cormode, D. Srivastava, N. Li, T. Li. Minimizing Minimality and Maximizing Utility: Analyzing Method-based attacks on Anonymized Data. In *VLDB*, 2010.

[5] G. Cormode, D. Srivastava, T. Yu, Q. Zhang. Anonymizing Bipartite Graph Data using Safe Groupings In *VLDB*, 2008.

[6] *http://en.wikipedia.org/wiki/Fisher-Yates_shuffle*

[7] A. Friedman and A. Schuster Data mining with differential privacy In *SIGKDD*, 2010.

[8] Samarati. Protecting respondents identities in microdata release. In *TKDE*, 2001.

[9] L. Sweeney. k-anonymity: a model for protecting privacy. In *International Journal on Uncertainty, Fuzziness and Knowledge-based systems*, 2002.

[10] G. Ghinita, Y. Tao, and P. Kalnis. On the anonymization of sparse high-dimensional data. In *ICDE*, 2008.

[11] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-Diversity: Privacy Beyond k-Anonymity. In ICDE, 2006.

[12] V. Karwa, S. Raskhodnikova, A. Smith, G. Yaroslavtsev. Private Analysis of Graph Structure. In *VLDB*, 2011.

[13] S. Kasiviswanathan, K. Nissim, S. Raskhodnikova and A. Smith. Analyzing Graphs with Node Differential Privacy. In *TCC*, 2013.

[14] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Incognito: Efficient Full-Domain K-Anonymity. In *SIGMOD*, 2005.

[15] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k- anonymity and l-diversity. In *ICDE*, 2007.

[16] A. Sala et. al Sharing Graphs using Differentially Private Graph Models In *IMC*, 2011.

[17] A. Serjantov and G. Danezis. Towards an Information Theoretic Metric for Anonymity. In *PETS*, 2002.

[18] C. Task and C. Clifton. What should we protect? Defining differential privacy for social network analysis. In *Social Network Analysis and Mining*, 2013.

[19] R. C. Wong, A. W. Fu, K. Wang, J. Pei. Minimality Attack in Privacy Preserving Data Publishing. In *VLDB*, 2007.

[20] R. Wong, J. Li, A. Fu, and K. Wang. (α, k)-anonymity: An enhanced k-anonymity model for privacy-preserving data publishing. In *SIGKDD*, 2006.

[21] C. Dwork. Differential Privacy In *ICALP*, 2006.

[22] X. Xiao and Y. Tao. Anatomy: Simple and effective privacy preservation. In *VLDB*, 2006.

[23] Y. Yang et. al. Differential Privacy in Data Publication and Analysis. In *SIGMOD*, 2012.

[24] Q. Zhang, N. Koudas, D. Srivastava, and T. Yu. Aggregate query answering on anonymized tables. In *VLDB*, 2007.

[25] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *ICDE*, 2008.