Privacy-preserving Publishing of Multi-level Utility Controlled Graph Datasets

BALAJI PALANISAMY, University of Pittsburgh, USA
LING LIU, Georgia Institute of Technology, USA
YANG ZHOU, Georgia Institute of Technology, USA
QINGYANG WANG, Louisiana State University, USA

Conventional private data publication schemes are targeted at publication of sensitive datasets either after k-anonymization process or through differential privacy constraints. Typically these schemes are designed with the objective of retaining as much utility as possible for the aggregate queries while ensuring the privacy of the individual records. Such an approach, though is suitable for publishing aggregate information as public datasets, is inapplicable when users have different levels of access to the same data. We argue that existing schemes either result in increased disclosure of private information or lead to reduced utility when some users have more access privileges than the others. In this paper, we present an anonymization framework for publishing large datasets with the goals of providing different levels of utility to the users based on their access privilege levels. We design and implement our proposed multi-level utility-controlled 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 each user privacy and access privilege levels.

Additional Key Words and Phrases: Data Privacy, Multi-level Privacy, Data Anonymization, Bipartite Graphs, Association datasets

ACM Reference Format:
Balaji Palanisamy, Ling Liu, Yang Zhou, and Qingyang Wang. 2017. Privacy-preserving Publishing of Multi-level Utility Controlled Graph Datasets. 1, 1, Article 1 (August 2017), 19 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Publishing data that contains sensitive information about individuals is an important problem. Such datasets may include medical information, patient records, census information or sales transactions made by customers. 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. Typically, such microdata is stored in a relational table with each record corresponding to an individual. 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

XXXX-XXXX/2017/8-ART1 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

(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. In general, the edges in such graphs are more sensitive than the nodes. For instance, while it may not be sensitive to publish the nodes that represent the list of drugs sold in the store or the patients that visit the store, publishing which patient bought which drugs can be highly sensitive. However, such data may be of high value and importance for a number of purposes. 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 initial focus on the problem of data anonymization has been on tabular data, via *k-anonymization* [9], [10] and subsequent variations [13], [23], [17]. However, a direct application of these techniques do not yield proper results on graph data. To this end, several techniques for graph anonymization has been proposed in the recent past. While a number of those 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 need not be true in practice. 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 anonymization 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. Concretely, this paper makes three original contributions. 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. Second, we present an order-preserving safe grouping technique for grouping nodes of the graph to support aggregate queries. In contrast to conventional graph grouping techniques that provide only approximate answers to aggregate queries, the proposed grouping techniques provide highly accurate results to aggregate queries involving node predicates. Third but not the least, 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, including our order-preserving safe grouping technique and its associated reversible node label permutation methods. Section IV presents our experimental evaluation on real large association graphs. We discuss related work in Section V and we conclude in Section VI.

2 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. In the context of association bipartite graphs, the key privacy concern is the association between the nodes (such as which patient had purchased which drug). The list of nodes and attributes such as the list of drugs sold by a pharmaceutical company or the list of movies in the rating dataset are in general not sensitive and rather publicly known information. Also, we note that these association graphs are quite sparse. Each patient buys only a very small subset of the set of all available drugs. Similarly, each researcher is a co-author on only a small subset of all the published papers. This makes the total number of edges in the graph to be quite small compared to the maximum possible edges. We show an example of such a bipartite graph in Figure 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 1(a) and 1(c) and the associations captured by the bipartite graph is shown in Figure 1. 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.

2.1 Multi-level Utility/ Privacy Model

In general, there are 4 types of queries that can operate in a large association graph dataset [6].

- 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, the number of sales on the most popular drug.
- Type 1 - queries involving attribute predicates on one side. E.g.: the average sales on antibiotic drugs, the average number of drugs purchased by customers 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.: which drugs did patient with patient ID '53253' buy, who are the buyers of the drug, 'Setraline' ?

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

2.1.1 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.

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

Table 1. Patients

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

Table 2. Drugs

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

Table 3. Associations

One naive approach to realize multi-level privacy is to produce different separate instances of anonymized version of the graph to different data users based on their data access privilege levels. However this approach suffers from two drawbacks: first, the multi-level privacy problem can not tackled by producing independent versions of the anonymized data. For example, if the anonymization for a lower privileged user is done without the knowledge of the anonymization performed for a higher privileged user, then it is possible that these two versions when combined (e.g., when a higher privileged user access a lower privileged version of the dataset) can leak information beyond what is leaked by the sum total of the information leaked by them individually. Also, the approach of releasing individual independent versions is prone to increased redundancy. For large scale datasets, such versions can significantly increase the storage cost for managing these independent redundant datasets. Next we discuss some existing privacy approaches for graph privacy protection and their applicability for the multi-level graph access problem considered in this work.

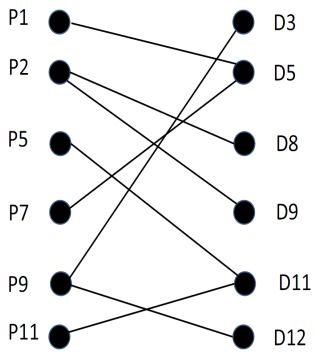


Fig. 1. Association Graph

2.2 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.

2.2.1 k-anonymity-based approaches. A large number of existing techniques anonymize data based on the concept of *k-anonymization* [9][10]. 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 (a) - Table (c)) for the patient-drug association graph shown in Figure 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 [9, 10]. 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 [6]. 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. Also, the tabular anonymization approach is shown to provide less useful query results in graph datasets as it does not preserve the graph structure [6]. We will discuss more examples of k-anonymity based graph anonymization techniques in Section 5.

Here, we note that the privacy strength of k-anonymized data can be challenged when the adversary has sources of background knowledge information about the individuals beyond the information contained in the published dataset. As anonymized data in general is expected to provide higher data utility than other techniques such as a differentially private data release, they are more preferable in scenarios where there is a clear lack of background knowledge information to the adversary or when the perceived risk of background knowledge attacks is minimal. However, in scenarios when the amount of background knowledge available to the adversary can be substantial, differential privacy techniques may provide a rigorous strength even though they are in general shown to provide less utility compared to k-anonymization. We discuss them next.

2.2.2 Differential Privacy-based data publication. An alternate approach to *k-anonymity*-based data publication is to release statistics of the dataset through differential privacy constraints [24]. It is believed that syntactic privacy notions are weaker and the state of the art differential privacy is becoming a more standard privacy notion. 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 [18]. More recent work on differential privacy for graph data has addressed node differential privacy [29, 30, 32] and in the context of social network graphs [32]. These 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.

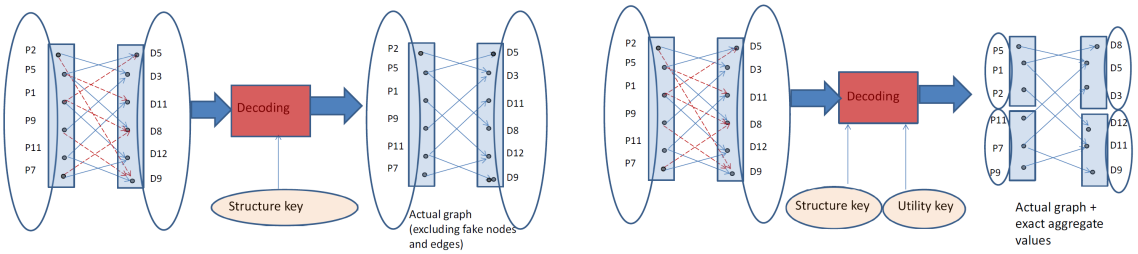
3 MULTI-LEVEL GRAPH ANONYMIZATION

This sections 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 (a). 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 (a)). 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 graph structure (Figure 2 (b)). In the same way, level 3 users use their association key to obtain access to exact associations in the graph (Figure 3), thereby obtaining the highest utility out of the dataset. We discuss the various steps involved in the anonymization process as follows:

3.1 Perturbing 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. We note that the goal of the perturbation algorithm is to only protect the utility of publishing the graph structure but as such exposing small parts of the unlabeled graph itself is not a privacy breach. In other words, if there are a few nodes with unique degrees and these degrees are known to the attacker, these nodes may be re-identified. But in a variety of examples, with just such small information, virtually nothing can be inferred about the total graph structure [6]. Therefore, the structure perturbation algorithm aims at protecting the overall graph structure from level 0 users and is not targeted to protecting every possible node and edge from an attacker, especially if the attacker has some prior knowledge about the unique degrees of a small number of nodes in the graph.



(a) Access for Level 1 users

(b) Access for Level 2 users

Fig. 2. Multi-level Data Access

3.1.1 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 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

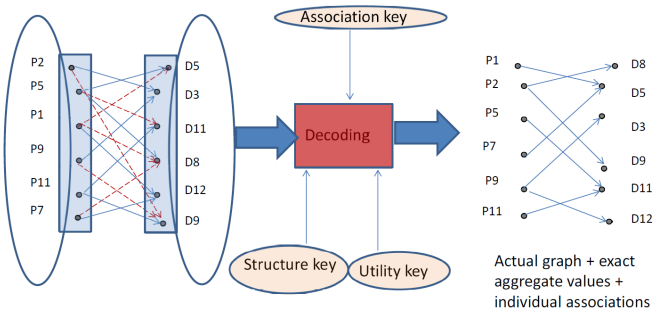


Fig. 3. Access for Level 3 users

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 detailed steps of this perturbation process is shown in Algorithm 1. We find that the worst case time complexity is $O(n \times |E|)$ where n fake edges are inserted and each fake edge collides with all existing edges in the graph. Here, we assume that the pseudorandom generator employs an $O(1)$ generator. Similarly, we find that the space complexity of the algorithm is $O(|E|)$ where $|E|$ represent the number of edges in the perturbed graph. The de-anonymization process works in a similar way (Algorithm 2). 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. From Algorithm 2, we note that the worst case time complexity of the algorithm is $O(n)$ where n fake edges are removed from the existing graph. Here again, the space complexity of the algorithm is $O(|E|)$ where $|E|$ represent the number of edges in the perturbed graph.

We note that prior to the decoding process, the data recipient needs to obtain the shared secret key. This is a typical instance of the key management problem that requires exchanging identical keys between the data publisher and data receiver. As directly exchanging symmetric keys in plain text would enable an adversary to intercept the key, the use of key management techniques [33, 34] would help protect the adversary from inferring the exchanged key. Additionally, the security of the key exchange procedure can be enhanced through the use of public key-based techniques. Extending the proposed framework using public-private key pairs is one potential direction of future work that may mitigate some of the key management issues associated with secret key sharing.

3.2 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. For instance, if the patient

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
F: a random seed generator function that returns the m^{th} pseudorandom number as a seed
procedure GraphPerturb(G, K_s, n)
    for $i = 1$ to n **do**
        $m = 0$
        $seed = F(K_s, m)$
        $R_2 i = PseudoRandom(2i, seed)$
        $R_{2i+1} = PseudoRandom(2i + 1, seed)$
        while $(R_{2i} mod |V|, R_{2i+1} mod |W|) \in E$ **do**
            $seed = F(K_s, m)$
            $R_2 i = PseudoRandom(2i, seed)$
            $R_{2i+1} = PseudoRandom(2i + 1, seed)$
            $m = m + 1$
        end while
        add edge $(R_{2i} mod |V|, R_{2i+1} mod |W|)$ to E
        if $m > 0$ **then**
            add $T(i, m)$
        end if
    end for
end procedure

nodes are ordered in the increasing order of age, then level 1 users can infer the results of aggregate queries for predicates involving age ranges. 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[7] shuffling algorithm to generate the permutation of the vertices. A description of this key-based permutation is shown in Algorithm 3. 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. From algorithm 3, we can deduce that the worst case time complexity of the algorithm is $O|V|$. Similarly the space complexity is $O|V|$ where $|V|$ represents the number of vertices in the graph. For decoding this random permutation (Algorithm 4), 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. Here again, the worst case time and space complexity of the algorithm is $O|V|$. 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

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
F: a random seed generator function that returns the m^{th} pseudorandom number as a seed
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

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**
        $R_i = PseudoRandom(i, K_u)$
        $j = R_i mod |V|$
        $Swap(V[i], V[j])$
    end for
end procedure

3.3 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. For instance, in Figure 2, the patient nodes $P_1, P_2, P_5, P_7, P_9, P_{11}$ form two groups of 3 nodes each. The node labels within the group are permuted so that the exact edges between the groups can not be inferred. Similarly, the drug nodes $D_3, D_5, D_8, D_9, D_{11}, D_{12}$ are grouped together and their labels 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 [6]. Formally, a k-grouping of a graph is defined as follows:

Algorithm 4 Reverse Permutation

```
1: V: Array of vertices
2: K_u: Aggregate utility key
3: R_i: i^th random number
4: procedure NodeShuffle(V, K_u)
5:     for i = 1 to |V| do
6:         R_i = PseudoRandom(i, K_u)
7:         j = R_i mod |V|
8:         Swap(V[i], V[j])
9:     end for
10: end procedure
```

Definition 3.1. 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, $\forall v \in V : \exists V_v \subseteq V : |V_v| \geq k \cup (\forall v_i \in V_v : H(v_i) = H(v))$ 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 so as to provide *k-anonymity*. In the past, safety criteria for the group formation has been studied and the notion of (k, l)-anonymity has been proposed [6]. When the nodes in V are grouped into groups of k or more nodes and if the nodes in W are also grouped into groups of l or more nodes, the result is what is referred to as a (k, l) grouping. However, existing grouping algorithms [3, 6] result in reordering of the nodes in order to achieve the safety condition which requires that any two nodes in the same group of V have no common neighbors in W . In such safe groups, given nodes $v \in V$ and $w \in W$ in groups of size k and l respectively, there are $max(k, l)$ possible identifications of entities with nodes and the edge (v, w) is in at most a $\frac{1}{max(k,l)}$ fraction of such possible identifications. Another security property of safe (k, l) grouping is that a grouping in which removing at most p nodes from V leaves a k-grouping of the remaining nodes of V , and removing at most q nodes from W leaves an l-grouping of the remaining nodes of W.

However, a major limitation of existing grouping techniques is the lack of node ordering which leads to providing only approximate answers to queries involving node predicates. For example, a query on finding the number of drugs purchased by customers who were born during the years 1972 - 1977 will typically involve nodes that are distributed over different groups. Therefore, an exact or very accurate answer to this query is not possible when the node ordering is lost. Also, the amount of error in the approximate answer increases with decrease in selectivity of the nodes based on the query predicate. As level 2 users in our problem setting have access to exact results of aggregate queries, such grouping schemes do not serve the purpose well.

Utility-preserving Safe Grouping: In order to facilitate retrieving exact aggregate query results from the grouped graph, we propose a utility-preserving safe grouping model called entropy-k safe grouping that provides the anonymity levels of conventional safe k grouping while enabling accurate results to be obtained for the aggregate queries. We begin by analyzing the privacy provided by conventional safe-k grouping in terms of the entropy [19] obtained from it. As conventional safe grouping ensures that for any given node, $w \in W$, there are no more than one edge to any given group of nodes in V , the probability, $p_{v_i,w}$ of associating a node v_i from a group of nodes V_{v_i} to w is given by

$$p_{v_i,w} = \sum_{v \in V_{v_i}} (p_{v \to v_i} * Y_{v,w})$$

where $p_{v,v_i,}$ represents the probability that label v_i is associated with node v and Y_{v_i,w_j} is a Boolean variable indicating if there is an edge between the nodes v and w.

Therefore, the entropy that captures the amount of information required to infer the associations of node w to nodes in the group V_v is given by

$$I(H_V(v), w) = -\sum_{v_i \in V_v} p_{v_i,w} \times log(p_{v_i,w})$$

We find that $p_{v_i,w} = \frac{1}{k}$ in a conventional safe k grouping and thus the safe k grouping has an entropy $I(H_V(v), w) \geq log(k)$ for all groups. In addition, conventional safe k grouping also has an additional property. If (k^p) is a grouping in which if at most p nodes are removed from V , then it leaves a k-grouping of the remaining nodes of V. Therefore, in the entropy k safe grouping as well, we would need to ensure that removing at most p nodes from V , leaves the entropy greater than or equal to $log(k-p)$ for the remaining nodes. The property holds good for any value of p in the range $[0, k]$. The intuition behind this property is to ensure that the inference of one edge (e.g., through external information such as when the user representing the patient node in a drug purchase association graph voluntarily reveals the drug purchased by her) in the grouped graph does not reduce the anonymity of the $k-$ grouped graph beyond that of a $k-1$ grouped graph. We formally define the notion of entropy k safe grouping as follows.

Definition 3.2. H_V is a Entropy k safe grouping of V in the context of a graph $G = (V, W, E)$, if the following condition holds:

1. $\forall w \in W, \forall v \in V$, the Entropy $I(H_V(v), w) \geq log(k)$

2. A k^p -grouping defined as a grouping in which removing at most p nodes from V leaves a grouping of the remaining nodes of V with Entropy $I(H_V(v), w) \geq log(k-p), \forall w \in W, \forall v \in V$.

Similar to an entropy k grouping of the nodes of V , an entropy k grouping of the nodes of W can be obtained. Next, we extend the notion of conventional safe (k, l) grouping to entropy safe (k, l) grouping. Consider a set of edges between a group of k vertices from V having edges that connect to nodes in a group of l vertices from W. Assume these sets of nodes are entropy k and entropy l grouped respectively. Given that the node labels are permuted in both the groups, an edge from node (v, w) indicates that the node v has k possible label mappings and similarly, the node w has l possible label mappings. Therefore, correctly inferring that there is an association (v, w) between the nodes v and w violates privacy.

The anonymity can be captured through entropy which is the amount of information required to infer that there is an edge (v, w) between the nodes v and w. Let $p_{v,w}$ be the probability that there is an edge between the node v and node w.

$$p_{v,w} = \sum_{v_i \in V_v} (p_{v \to v_i} * \sum_{w_j \in W_w} p_{v_i,w_j} * p_{w \to w_j})$$

$$I(v, w) = -\sum_{v_i \in V_v,\, w_j \in W_w} (p_{v \to v_i} * p_{v_i,w_j}) \times log(p_{v,v_i} * p_{v_i,w_j})$$

where p_{v,v_i} represents the probability of associating node v with label v_i and p_{v_i,w_j} represents the probability of having an association from node v_i to node w_j. In other words, $p_{v,v_i} * p_{v_i,w_j}$ is the probability of inferring that the edge (v, w) is (v_i, w_j). For a group containing k nodes, we know that p_{v,v_i} is $\frac{1}{k}$. Similarly, we find that p_{w,w_j} is at most $\frac{1}{l}$. Therefore, in a conventional safe (k, l) grouping, the probability of correctly inferring an edge is $\frac{1}{k*l}$. Therefore, when we try to preserve the ordering of the nodes, we need to ensure that the probability, $p_{v,w} \leq \frac{1}{k*l}$ for all groups in the grouped graph. In other words, this ensures that the entropy, $I(v, w)$ of each group is at least as high as that of the safe (k, l) grouping.

Definition 3.3. H_V is an order-preserving safe (k, l) grouping of V in graph $G = (V, W, E)$, if the following conditions hold:

1. $\forall v \in V_v$ and $\forall w \in W$, $p_{v,w} \leq \frac{1}{k*l}$

2. For any member $(v_i, v_j) \in P_v$, where P_v is the original ordering of V, $\exists (v_i, v_j) \in P'_v$, where P'_v is the ordering in the grouped graph of V.

3. For any member $(w_i, w_j) \in P_w$, where P_w is the original ordering of W, $\exists (w_i, w_j) \in P'_w$, where P'_w is the ordering in the grouped graph of W.

4. A $(k,l)^{p,q}$ -grouping defined as a grouping in which at most p nodes are removed from V and at most q nodes are removed from W, leaves a (k,l) grouping of the remaining nodes of V and W with $p_{v,w} \leq \frac{1}{k*l}$, $\forall w \in W, \forall v \in V$.

The intuition behind conditions 2 and 3 in the definition is to ensure that the nodes in the grouped graph retain the ordering so that range queries based on the ordering attribute can be answered with higher accuracy. The intuition behind condition 4 is to ensure that the inference of one edge in a (k,l) grouped graph does not reduce the anonymity of the (k,l) grouped graph beyond that of a $(k-1, l-1)$ grouped graph. Based on these assumptions, we design our order-preserving grouping algorithm that ensures the same level of entropy as that of conventional safe grouping and yet preserves node ordering to provide highly accurate aggregate query answers.

3.3.1 Order-preserving Grouping Algorithm. The idea behind the order-preserving grouping algorithm is to divide the sets of given vertices, V and W into disjoint groups of nodes such that the nodes in the list of grouped nodes follows the original order of V and W respectively and each pair of groups preserves the order preserving safe (k,l) grouping anonymity requirements. A straight-forward way to implement such a grouping would be to order the nodes in the desired order and start adding nodes to a new group in the same order until the group satisfies the entropy (k,l) grouping properties. Once a group is formed, the algorithm proceeds to start adding nodes to the next group in order. Although such a straight-forward algorithm might be intuitively trying to achieve an order-preserving safe grouping, we note that they are subject to minimality attacks [5, 22]. In general, an anonymziation scheme that aims at minimizing the anonymity group size to merely guarantee the required anonymity is prone to minimality attacks. It has been shown in prior work[5] that one way to develop a minimality-attack resilient anonymization technique is to design an even split grouping algorithm [5] that partitions the set of nodes into hierarchical partitions and then exposes only those partitions that meet the safe grouping anonymity requirements. As the partitions and partition sizes in an even split approach are pre-determined and independent of the input data, the partitions are not chosen based on minimalistic partition size to just meet anonymity requirements and hence, such an approach is naturally resilient to minimality attacks. We refer the interested readers to [5, 22] for additional details on minimality attack and how even split partitioning avoids minimality attack.

Our order-preserving group formation algorithm is designed as an even-split algorithm. The proposed algorithm (Algorithm 5 begins by recursively partitioning the nodes V and W into two sets of groups $G(v)$ and $G(w)$ where $G(v) = \{G_1^v, G_2^v, ..., G_x^v\}$ such that $\{G_1^v, G_2^v, ..., G_x^v\}$ is a partition of V and $G(w) = \{G_1^w, G_2^w, ..., G_y^w\}$ such that $\{G_1^w, G_2^w, ..., G_y^w\}$ is a partition of W . Each group G_i^v where $(1 \leq i \leq x)$ and each group G_i^w where $(1 \leq i \leq y)$ satisfies (k,l) order-preserving grouping property. Initially there is only one group G_1^v. As the grouping algorithm continues to partition the graph and forms groups, additional groups such as G_2^v, G_2^v and G_3^v are created and added to $G(v)$. These groups are similar to the groups $\{p_5, p_1, p_2\}$ and $\{p_1 1, p_7, p_9\}$ shown in Figure 2(b). In Algorithm 5, $G'(v)$ and $G'(w)$ are local variables and are similar to $G(v)$ and $G(w)$.

Initially, the two global sets of partitioned groups start with $G(v) = V$ and $G(w) = W$ respectively. In other words, the set of nodes V represents one group and W represents the other group. The splitting algorithm starts initially with $G_v = V$ and $G_w = W$ respectively. It splits $G_v = \{v_p, v_{p+1}, v_{p+2}, ...v_{p+r}\}$ into $G_v^{m_1} = \{v_p, v_{p+1}, v_{p+2}, ...v_{\frac{p+r}{2}}\}$ and $G_v^{m_2} = \{v_{\frac{p+r}{2}+1}, v_{\frac{p+r}{2}+2}....v_r\}$ where $r = |G_v|$. Similarly, it splits $G_w = \{v_q, v_{q+1}, v_{q+2}, ...v_{q+s}\}$ into $G_v^{n_1} = \{v_q, v_{q+1}, v_{q+2}, ...v_{\frac{q+s}{2}}\}$ and $G_v^{n_2} = \{v_{\frac{q+s}{2}+1}, v_{\frac{q+s}{2}+2}....v_s\}$ where $s = |G_w|$.

Algorithm 5 Order-preserving Grouping

V: Array of vertices representing first set of nodes
W: Array of vertices representing second set of nodes
Input: V and W and parameters k and l.
Output: Two set of groups represented by global variables,$G(v)$ and $G(w)$ where $G(v) = \{G_1^v, G_2^v, ..., G_x^v\}$ such that $\{G_1^v, G_2^v, ..., G_x^v\}$ is a partition of V and $G(w) = \{G_1^w, G_2^w, ..., G_y^w\}$ such that $\{G_1^w, G_2^w, ..., G_y^w\}$ is a partition of W . Each group G_i^v where $(1 \leq i \leq x)$ and each group G_i^w where $(1 \leq i \leq y)$ satisfies (k, l) order-preserving grouping property. Initially there is only one group G_1^v. As the grouping algorithm continues to partition the graph and forms groups, additional groups such as G_2^v, G_2^v and G_3^v are created and added to $G(v)$. $G'(v)$ and $G'(w)$ are local variables and are similar to $G(v)$ and $G(w)$.
procedure OrderPreservingGrouping(V, W, k, l)
    $G_1^v = V$
    $G_1^w = W$
    $G(v) = \{G_1^v\}$
    $G(w) = \{G_1^w\}$
    Split (G_1^v, G_1^w, k, l)
end procedure
procedure Split(G_v, G_w, k, l)
    Split the current group G^v into two halves and G^w into two halves
    $r = |G_v|$
    $s = |G_w|$
    Split $G_v = \{v_p, v_{p+1}, v_{p+2}, ...v_{p+r}\}$ into $G_v^{m_1} = \{v_p, v_{p+1}, v_{p+2}, ...v_{\frac{p+r}{2}}\}$ and $G_v^{m_2} = \{v_{\frac{p+r}{2}+1}, v_{\frac{p+r}{2}+2}....v_r\}$
    Split $G_w = \{v_q, v_{q+1}, v_{q+2}, ...v_{q+s}\}$ into $G_v^{n_1} = \{v_q, v_{q+1}, v_{q+2}, ...v_{\frac{q+s}{2}}\}$ and $G_v^{n_2} = \{v_{\frac{q+s}{2}+1}, v_{\frac{q+s}{2}+2}....v_s\}$
    $G'(v) = G(v) - G_v \cup G_v^{m_1} \cup G_v^{m_2}$
    $G'(w) = G(w) - G_w \cup G_w^{n_1} \cup G_w^{n_2}$
    if Each group in ($G'(v)$ and $G'(w)$ satisfy order preserving safe (k, l) grouping) **then**
        $G(v) = G'(v)$
        $G(w) = G'(w)$
        Split ($G_v^{m_1}, G_w^{n_1}, k, l$)
        Split ($G_v^{m_2}, G_w^{n_2}, k, l$)
    else
        return
    end if
end procedure

The algorithm then checks if the set of groups formed after the splitting process satisfies the safe (k, l) grouping requirements. If the safe (k, l) grouping requirements are met, the splitting algorithm recursively splits the newly formed groups $G_v^{m_1}$ and $G_w^{n_1}$ and $G_v^{m_2}$ and $G_w^{n_2}$. This recursive splitting process proceeds until no groups in $G(v)$ and $G(w)$ can be split further. We can see that in the worst case, when $k = 1$ and $l = 1$, requiring no anonymity, the number of splits/ groups formed by the partition would be equal to $max(|V|, |W|)$, the maximum of the number of nodes in V and W. As each split operation would incur a test of safe (k, l) grouping requirement for the pair of newly formed sub groups, it would incur a worst case time complexity of $max(|V|, |W|) \times max(|V|, |W|)$ (based on the definition of $I(v, w)$) to check for the safe (k, l) grouping criterion at each step. Thus the overall worst case time complexity of the algorithm is $max(|V|, |W|)^3$.

4 EXPERIMENTAL EVALUATION

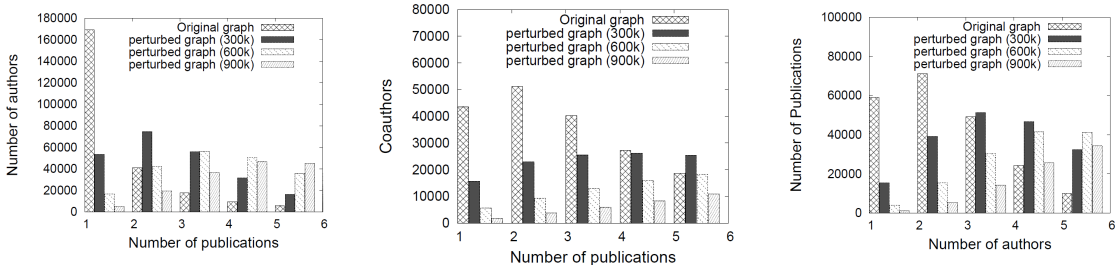
Our experimental evaluation consists of evaluating both privacy as well as performance efficiency of the proposed anonymization schemes. We first evaluate the effectiveness of the proposed techniques in terms of the privacy protection by measuring query accuracy and privacy levels offered by the proposed order-preserving safe grouping approach. We then evaluate the performance of the proposed key-based anonymization schemes in terms of anonymization and de- anonymization time and space efficiency for various privacy and utility levels. Before we present our experimental results, we first describe our experimental setup including the real dataset used in the experiments.

The proposed anonymization and de-anonymization schemes are implemented in Java. We use an experimental setup similar to the one adopted in [6]. 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. The edges represent the associations that follow a power law distribution representing a sparse association graph.

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 the publication y?

We study the privacy and performance of the above queries for various access privilege levels by varying a number of other parameters such as group size, the degree of graph structure perturbation, selectivity of the query predicates. To measure accuracy of each query, the lower bound estimation L and the upper bound estimation U are computed. We also compute the expected value μ. If the correct answer to the query is Q, we compute two error measurements: the error bounds $\frac{U-L}{2Q}$ (the worst case error from using $(U + L)/2$ as an estimate for Q), and the expected error $\frac{|\mu-Q|}{Q}$.



(a) Distribution of No of papers (b) Distribution of collaborators (c) Distribution of Publications

Fig. 4. Effect of Graph Structure perturbation

Distributions	Original Vs 300k	Orignal Vs 600k	Original Vs 900k
No. of. Authors Vs No. of Publications	215258	588956	866687
Coauthors Vs No. of. Publications	133690	362189	526885
No. of. Publications Vs No. of. Authors	137384	449909	819032

Table 4. KL Divergence

4.1 Effect of Graph Structure Perturbation

Our first set of experiments study the protection provided by the key-based graph perturbation techniques on the utility for level 0 users. Note that level 0 users do not have access to the graph structure related queries. 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 5(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. We show the KL-Divergence of the distributions in Figure 4(a) - 4(c) in Table IV where we find that the KL-Divergence in general increases with increase in the number of fake edges added to the perturbed graph. Therefore, one way to decide the degree of perturbation is by increasing the number of fake edges in the perturbed graph so as to reach a required threshold of the KL-Divergence value.

Similarly, Figure 5(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 5(c).The perturbed graphs again show that level 0 users do not obtain accurate results to this distribution.

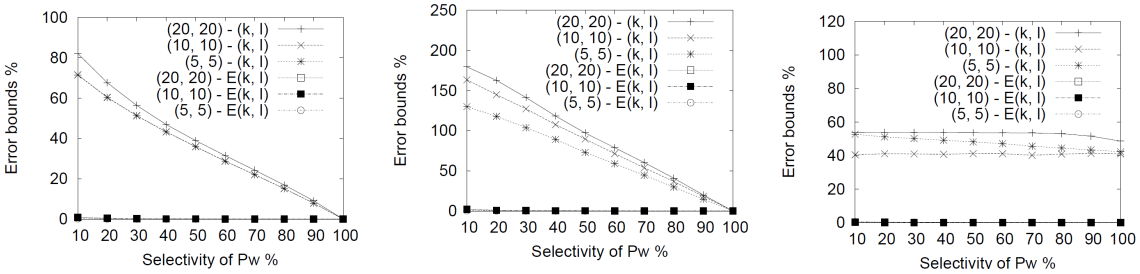
4.2 Effect of Order-preserving Safe Grouping

The next set of experiments studies the utility and privacy protection provided by the order-preserving safe (k, l) grouping technique for level 2 users. In Figure 6, we present the error bounds of two type 1 queries (Query A and Query B) and the type 2 query described in the experimental setup. We compare the error bounds of the proposed order-preserving (k, l) grouping with strict safe (k, l) grouping [6]. Figure 6(a) shows the comparison of the error bounds of Query A for safe (k, l) grouping and order-preserving (k, l) grouping for various values of k and l. On X-axis, the selectivity of the query, P_w is varied. We notice that safe (k, l) lacks ordering and hence leads to high error bounds, especially at lower selectivity levels. Also, the error bounds are higher when the group size is larger, when the k and l are larger. On the other hand, order preserving (k, l) grouping achieves high query accuracy and close to 0% error bounds. Such low error bounds enable aggregate query users to obtain accurate results to aggregate queries while the same results are protected against graph structure only users. In Figure 6(b), we present the error bounds for Query B which again shows that safe (k, l) grouping has higher errors at lower selectivity ranges. However, order-preserving (k, l)grouping achieves close to 0% error bounds at all selectivity values. Similarly, Figure 6(c) shows the error bounds for type 2 query discussed in the experimental setting. Here, the selectivity, P_w is held as a constant at 0.8 and the selectivity, P_w is varied along the X-axis. We find that safe (k, l) grouping has significant amount of errors for all values of P_w , however, the order-preserving (k, l) grouping has very low error bounds. Hence, aggregate query users obtain accurate results for these aggregate

queries. We note that the proposed order preserving algorithm preserves the ordering based on a single attribute in the published dataset. Therefore queries based on multiple attributes in our approach perform similar to the conventional (k, l) grouping approach as the order-preserving grouping does not protect ordering beyond a single attribute.

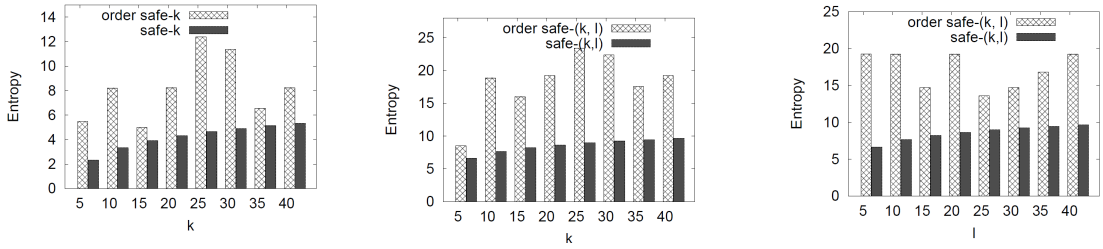
4.3 Privacy of Order-preserving Safe Grouping

We next compare the privacy of order-preserving safe grouping with conventional safe grouping based on the level of entropy provided by them. Here, the entropy captures the amount of information required to infer the associations of nodes in a group to another group. Figure 7(a) shows the average entropy of a k-grouped graph using both conventional safe-k grouping and order-preserving safe-k grouping. Here, only the author nodes in the graph are considered for grouping. We note that the order-preserving safe-k grouping has an entropy at least as high as the safe-k grouping and on an average, we find that order-preserving grouping has a much higher entropy than the safe-k grouping for various values of k, the group size. Next, we consider grouping both author nodes and publication nodes in a (k, l) grouping. We keep $l = 20$ for the group size on publication nodes and vary the size of the author node groups, k. We find that the order-preserving (k, l) grouping provides a significantly higher entropy than the conventional (k, l) grouping. This graph basically shows an empirical evidence of the theoretical property of the order-preserving (k, l) grouping that ensures that each such group provides an entropy at least as high as that of a safe (k, l) grouping. Figure 7(c) shows a similar observation where the group size of the author nodes, k is fixed as 20 and the group size of publication nodes, l is varied on the X-axis.



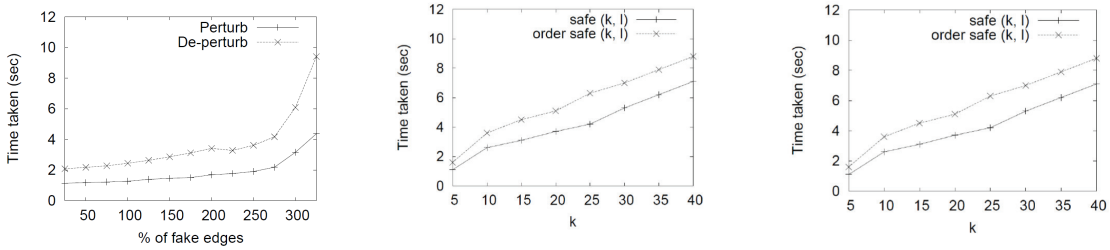
(a) Type 1: Query A (b) Type 1: Query B (c) Type 2 Query

Fig. 5. Effect of Order-preserving Safe Grouping



(a) Safe k grouping (b) (k, l) grouping with l = 20 (c) (k, l) grouping with k = 20

Fig. 6. Privacy of Order-preserving Safe Grouping



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

Fig. 7. Performance of perturbation and grouping techniques

4.4 Performance of Perturbation and Grouping techniques

Our final 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 8(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 of 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 8(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 order preserving grouping algorithm takes only minimal additional time compared to the conventional safe grouping technique. Similarly, the time taken for the node label permutation operation in Figure 8(c) indicates that the process is quite fast and scales well for various group sizes.

4.5 Space Efficiency

We compare the space efficiency of the proposed multi-level privacy protection approach with a straightforward approach of publishing separate instances of the dataset for each privacy level. In Figure 8, we compare the total size of the published datasets for the proposed approach with the straightforward approach. We compare three scenarios namely (i) a data publishing scenario requiring the protection of all three privacy levels, level 1, 2 and 3, (ii) requiring only the protection of level 1 and 2 and (iii) requiring only the protection of level 3. We note that the size of the published datasets increase significantly in the straight forward approach as the number of privacy levels increase. In contrast, the proposed approach minimizes the redundancy and has a higher space efficiency in terms of dataset size for all the three scenarios.

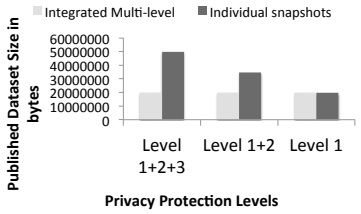


Fig. 8. Comparison of Dataset size

5 RELATED WORK

The problem of information disclosure has been studied extensively in the framework of statistical databases. Samarati and Sweeney [9, 10] introduced the k-anonymity approach which has led to new techniques and definitions such as l-diversity [13], (α, k)-anonymity [23],t-closeness [17] and anonymization via permutation [25, 27]. However, these schemes are

primarily targeted for data publishing with the goal to provide aggregate queries while protecting individual information. There had been some work on anonymizing graph datasets with the goal of publishing statistical information without revealing information of individual records. Backstrom et al. [2] show that in fully censored graphs where identifiers are removed, a large enough known subgraph can be located in the overall graph with high probability. In [28], 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. A more sophisticated attack is discussed in [12] where the attacker has the ability to buy information about the neighborhood of certain nodes. Ghinita et al. present an anonymization scheme for anonymizing sparse high-dimensional data using permutation based methods [11] 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 [3, 6] 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. Hence, these techniques do not handle scenarios when different users have different levels of access to the same data.

A state of the art direction of privacy research is represented by differential privacy techniques. Based on the concept of differential privacy introduced in [24], there had been many work focused on publishing aggregates of sensitive datasets through differential privacy constraints [4, 8, 26]. 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 [14, 15, 20]. 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 [18]. More recent work on differential privacy for graph data has addressed node differential privacy [29, 30, 32] and in the context of social network graphs [32]. 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. A promising direction of future work is to extend the concepts and principles developed in this work to release differentially private multi-level privacy protected graph data. While the core techniques and principles developed in this work will still continue to apply, concepts and research results from differentially private graph release techniques such as [29–32] may be employed to ensure differential privacy guarantees in the published multi-level access controlled data.

6 CONCLUSION

This paper presents an anonymization framework for publishing large association graph datasets with the goal of supporting multi-level access controlled query processing. 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 a wide range of 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.

7 ACKNOWLEDGEMENT

This research is partially supported by the National Science Foundation under Grants NSF 1547102, SaTC 1564097 and IBM faculty award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or IBM.

REFERENCES

[1] C. Aggarwal. On k-Anonymity and the Curse of Dimensionality. In *VLDB*, 2005.
[2] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore are thou R3579X? Anonymized social networks, hiddern patterns and structural steganography. In *WWW*, 2007.
[3] S. Bhagat, G. Cormode, B. Krishnamurthy, D. Srivastava. Class-based graph anonymization for social network data. In *VLDB*, 2009.
[4] R. Chen Publishing Set-Valued Data via Differential Privacy. In *VLDB*, 2011.
[5] G. Cormode, D. Srivastava, N. Li, T. Li. Minimizing and Maximizing Utility: Analyzing Method-based attacks on Anonymized Data. In *VLDB*, 2010.
[6] G. Cormode, D. Srivastava, T. Yu, Q. Zhang. Anonymizing Bipartite Graph Data using Safe Groupings In *VLDB*, 2008.
[7] *http://en.wikipedia.org/wiki/Fisher-Yates_shuffle*
[8] A. Friedman and A. Schuster Data mining with differential privacy In *SIGKDD*, 2010.
[9] Samarati. Protecting respondents identities in microdata release. In *TKDE*, 2001.
[10] L. Sweeney. k-anonymity: a model for protecting privacy. In *International Journal on Uncertainty, Fuzziness and Knowledge-based systems*, 2002.
[11] G. Ghinita, Y. Tao, and P. Kalnis. On the anonymization of sparse high-dimensional data. In *ICDE*, 2008.
[12] A. Korolova, R. Motwani, S. Nabar, and Y. Xu. Link privacy in social networks. In *ICDE*, 2008.
[13] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-Diversity: Privacy Beyond k-Anonymity. In ICDE, 2006.
[14] V. Karwa, S. Raskhodnikova, A. Smith, G. Yaroslavtsev. Private Analysis of Graph Structure. In *VLDB*, 2011.
[15] S. Kasiviswanathan, K. Nissim, S. Raskhodnikova and A. Smith. Analyzing Graphs with Node Differential Privacy. In *TCC*, 2013.
[16] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Incognito: Efficient Full-Domain K-Anonymity. In *SIGMOD*, 2005.
[17] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k- anonymity and l-diversity. In *ICDE*, 2007.
[18] A. Sala et. al Sharing Graphs using Differentially Private Graph Models In *IMC*, 2011.
[19] A. Serjantov and G. Danezis. Towards an Information Theoretic Metric for Anonymity. In *PETS*, 2002.
[20] C. Task and C. Clifton. What should we protect? Defining differential privacy for social network analysis. In *Social Network Analysis and Mining*, 2013.
[21] G. Toth, Z. Hornak and F. Vajda. Measuring Anonymity Revisited. In *Norsec*, 2004.
[22] R. C. Wong, A. W. Fu, K. Wang and J. Pei. Attack in Privacy Preserving Data Publishing. In *VLDB*, 2007.
[23] 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.
[24] C. Dwork. Differential Privacy In *ICALP*, 2006.
[25] X. Xiao and Y. Tao. Anatomy: Simple and effective privacy preservation. In *VLDB*, 2006.
[26] Y. Yang et. al. Differential Privacy in Data Publication and Analysis. In *SIGMOD*, 2012.
[27] Q. Zhang, N. Koudas, D. Srivastava, and T. Yu. Aggregate query answering on anonymized tables. In *VLDB*, 2007.
[28] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *ICDE*, 2008.
[29] W. Day, Ni. Li and M.Lyu. Publishing Graph Degree Distribution with Node Differential Privacy. In *SIGMOD*, 2016.
[30] J. Zhang, G. Cormode, C. Procopiuc, D. Srivastava, X. Xiao. Private release of graph statistics using ladder functions. In *SIGMOD*, 2015
[31] J. Blocki, A. Blum, A. Datta and O. Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *ITCS*, 2013
[32] S. Chen and S. Zhou. Recursive mechanism: Towards node differential privacy and unrestricted joins. In *SIGMOD*, 2013
[33] E. Barker et al. NIST Special Publication 800 -130: A Framework for Designing Cryptographic Key Management Systems. In *National Institute of Standards and Technology Report*, 2016.
[34] D. Turner. "What Is Key Management? A CISO Perspective". In *Cryptomathic. Retrieved 30 May 2016.*