The VLDB Journal manuscript No.
(will be inserted by the editor)

Anonymizing Bipartite Graph Data using Safe Groupings

Graham Cormode · Divesh Srivastava · Ting Yu · Qing Zhang

Received: date / Accepted: date

Abstract Private data often comes in the form of associations between entities, such as customers and products bought from a pharmacy, which are naturally represented in the form of a large, sparse bipartite graph. As with tabular data, it is desirable to be able to publish anonymized versions of such data, to allow others to perform ad hoc analysis of aggregate graph properties. However, existing tabular anonymization techniques do not give useful or meaningful results when applied to graphs: small changes or masking of the edge structure can radically change aggregate graph properties.

We introduce a new family of anonymizations for bipartite graph data, called (k, ℓ)-groupings. These groupings preserve the underlying graph structure perfectly, and instead anonymize the mapping from entities to nodes of the graph. We identify a class of "safe" (k, ℓ)-groupings that have provable guarantees to resist a variety of attacks, and show how to find such safe groupings. We perform experiments on real bipartite graph data to study the utility of the anonymized version, and the impact of publishing alternate groupings of the same graph data. Our experiments demonstrate that (k, ℓ)-groupings offer strong tradeoffs between privacy and utility.

Keywords Privacy · Microdata · Graph · Query Answering

Yu and Zhang were partially sponsored by the NSF through grants IIS-0430166 and CNS-0747247.

Graham Cormode
AT&T Labs–Research, Florham Park, NJ
E-mail: graham@research.att.com

Divesh Srivastava
AT&T Labs–Research, Florham Park, NJ
E-mail: divesh@research.att.com

Ting Yu
North Carolina State University, Raleigh, NC
E-mail: tyu@ncsu.edu

Qing Zhang
North Carolina State University, Raleigh, NC
E-mail: qzhangqing@gmail.com

Present address:
Qing Zhang
Teradata, El Segundo, CA

1 Introduction

Private data often arises in the form of *associations* between entities. A first example is represented by the products bought by customers at a pharmacy. The set of products being sold and their properties is public knowledge, and it may be no secret which customers visit a particular pharmacy. However, the association between a particular individual and a particular medication is often considered sensitive, since it is indicative of a disease or health issue that they have. A second example of association data is the Netflix prize data set, released in 2006, which was anonymized based on an unspecified heuristic method [2]. This led to speculation on how easy it would be to break the privacy [13]. A third example is that of authors and papers: for a conference such as SIGMOD, reviewers learn information about submitted papers (title, area, abstract), and could (in future) also see detailed information about authors who have submitted papers, in order to verify conflicts of interest. But, since SIGMOD is a double-blind conference, the association between authors and papers should not be revealed to reviewers.

The most natural way to model such data is as a graph structure: nodes represent entities, and edges indicate an association between them; much analysis can then be performed on structural properties of this graph. In this work, we study data that can be modeled as bipartite graphs—there are two types of entity, and associations link together one entity of each type. In the pharmacy, customers buy products, and in SIGMOD, authors write papers, building (customer, product) and (author, paper) associations respectively. Each entity can be involved in few or many associations, but in most common situations, only a tiny fraction of all possible associations are present. No customer buys more than a small fraction of the available products, and no product is bought by more than a small fraction of the total customers. Similar observations hold for publication data about authors and the papers they have written. In other words, the induced graph is quite sparse, and we must ensure that these associations are not easily revealed.

Although the data is private, it is still desirable to allow aggregate analysis based on the structure of the graph. Pharmaceutical companies wish to understand which pattern of products are bought by people in particular age ranges; public health organizations want to watch for disease outbreaks affecting particular demographics based on certain types of medicine being purchased; SIGMOD may encourage analysis of hot topics in databases, or better understanding of coauthorship patterns. Publishing the raw data would allow these questions to be answered directly, but would fail to meet the privacy concerns outlined above. The model where the data owner accepts queries and either adds noise to results or refuses to answer some questions requires the data owner to be an active participant and may limit what analysis is possible. Instead, we adopt the approach of publishing some anonymized version of the data, and ensuring that the scope for inferring any given association from this data is limited while the key properties, in particular the structure of the underlying graph, are preserved. This approach allows a wide variety of ad hoc analyses and novel valid uses of the data, while ensuring our privacy goals are met.

The problem of publishing anonymized data has attracted significant interest in recent years [10–12, 16, 18, 20, 23]. However, the focus has mostly been on tabular data, rather than the associations we study here. As a consequence, applying existing anonymization techniques tends to erase almost all structure, so that little use can be made of the resulting data. Moreover, a tabular approach ignores the inherent graph properties which hold a lot of the value of the data: e.g. structure such as number of customers buying the same product, collaboration "hop" distance between a pair of authors, pattern of other common products between customers using the same product, and so on. These are all important features of interest for aggregate analysis, but are radically altered by simply treating the data as a table and masking or perturbing the data. In Section 3, we work through several detailed examples to show that existing approaches for tabular data are insufficient for anonymizing associations.

Some recent work has begun to address anonymizing graph data, motivated by the structures present in social network data. But rather than proposing ways to modify the pattern of links in the graph to ensure privacy, our work differs by making different assumptions about the strength of the attacker and the utility of the graph data. Prior work [1, 8] tends to assume a lot of knowledge or power on behalf of the attacker (in particular, knowledge of node degrees, or of particular subgraphs, and the ability to insert new nodes and edges into the graph), and shows that under such assumptions some associations can be inferred. In contrast, we address a different but equally important range of the privacy-utility tradeoff. We give a new approach for anonymizing associations which can be represented as bipartite graphs and show it to be resilient against certain attack models.

1.1 Our Contributions.

Our methodology is based on the idea that rather than masking or altering the graph structure, we should preserve the graph structure exactly, and instead focus on masking the mapping from entities to nodes of the graph. This approach ensures that the complex and sensitive graph structure is not affected, and so we can be sure that any analysis based principally on the graph structure will be correct. Privacy is ensured by *grouping* the nodes and entities: we partition the nodes in the graph, and the corresponding entities, into groups so that, given a group of nodes, there is a (secret) mapping from these nodes to the corresponding group of entities. There is no information published that would allow an attacker to work out, within a group, which node corresponds to which entity. This gives a tradeoff between privacy and utility: intuitively, larger groups give more privacy, but less certainty when answering queries which select a subset of entities.

We give a simple condition for a grouping to be *safe*, which precisely limits the ability of an attacker to make any inference from the published information alone. We provide an algorithm which is successful at finding safe groupings in practice, and go on to describe how to answer a variety of query types efficiently given the published anonymized data. We also give formal analysis of how little can be deduced by an attacker who has additional background knowledge in the form of known associations between particular pairs of entities, and show that there is high security for entities about whom no information is known by the attacker.

We demonstrate the efficacy of our approach with a careful experimental analysis of the ease of building safe groupings, and the accuracy with which a variety of queries can be answered over such anonymized data. We also study the effect of variations of our approach, and demonstrate that techniques based on publishing two versions of the same data, while significantly increasing the utility and accuracy of query answering, can also expose more associations to unintended revelation.

Customer	State
c1	NJ
c2	NC
c3	CA
c4	NJ
c5	NC
c6	CA

(a) Customer table

Product	Availability
p1	Rx
p2	OTC
p3	OTC
p4	OTC
p5	Rx
p6	OTC

(b) Product table

Customer	Product	Customer	Product
c1	p2	c4	p5
c1	p6	c5	p1
c2	p3	c5	p5
c2	p4	c6	p3
c3	p2	c6	p6
c3	p4		

(c) Customer-Product table

Fig. 1 Example data set in tabular form

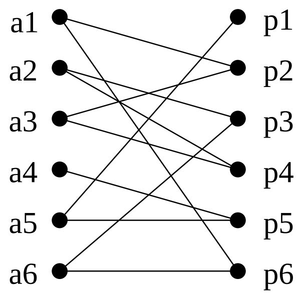


Fig. 2 Example data set in graph representation

Outline. The paper proceeds as follows: We describe the data model, privacy model, query model and experimental model in Section 2. We show explicitly that prior work on tabular data fails to give useful results when applied to the kind of data that we study in Section 3. We propose our new approach based on grouping in Section 4, and analyze its properties, proving security against a natural class of attacks, and giving experimental evidence of its utility. We consider variations based on publishing multiple anonymizations of the same data in Section 5, and analyze the resulting privacy. Related work is reviewed in Section 6, and concluding remarks are given in Section 7.

2 Preliminaries

2.1 Graph Model

Throughout, we focus on problems of anonymizing bipartite graphs $G = (V, W, E)$ (bigraph for short). That is, the bigraph G consists of $m = |V|$ nodes of one type, $n = |W|$ nodes of a second type, and a set of $|E|$ edges $E \subseteq V \times W$. Such graphs can encode a large variety of data, in particular, the set of existing links between two sets of objects. For example, we can encode which papers were co-written by a set of authors; which products at a pharmacy were bought by a set of customers; which websites were visited by users; which courses were taken by students; and so on. Throughout, we shall work with an illustrative example of a set of customers $C = V$ and a set of products $P = W$. An edge (c, p) indicates customer $c \in C$ bought product $p \in P$. Observe that here, as in many of the examples above, the graph is relatively *sparse*: each customer typically buys only a small fraction of all products, and each product is bought by only a few customers (with a few exceptions, e.g. many customers buy aspirin). As a consequence, the number of edges e is small compared to the number of *possible* edges, which is $n \times m$. More formally, we say that a graph is α-sparse if $e \leq \alpha nm$; we will subsequently provide a necessary bound on the α-sparseness for our method to succeed. A second measure of sparseness looks at the degree of each node: a graph is β-sparse if the maximum degree of a node in W is at most βm and the maximum degree of a node in V is at most βn. In full generality, we can consider directed graphs with multiedges, with weights or additional attributes. However, for clarity, we describe only the unweighted, undirected, single edge case: this has sufficient richness to capture many challenging problems.

In a relational database, a bipartite graph $G = (V, W, E)$ is naturally and concisely represented by three tables, corresponding to V, W and E. In our example, we would have a table of customers V, including attributes such as gender and location (from a customer loyalty scheme, say); a table of products W, including attributes such as price, type, and whether it is available Over the Counter (OTC) or by Prescription Only (Rx); and a customer-product table E encoding who bought what. Thus *entities* in the tables V and W correspond to *nodes* in the graph defined by E, in a 1:1 fashion.

Example 1 Figure 1 shows a sample instantiation of this schema with Figure 2 showing the graph representation of the customer-product relation in Figure 1(c). The α sparsity of this graph is 11/36 (there are 11 edges present out of 36 possible edges), and the β sparsity is 1/3 (no node is connected to more than 2 out of the 6 possible nodes). Customers have an additional attribute, state, indicating whether they are based in New Jersey (NJ), North Carolina (NC) or California (CA). The availability of a product indicates whether it is Over the Counter or Prescription Only. Since the graph accurately represents the relational data, we use both graph and relational terminology. $\square$

2.2 Privacy Goals

Our objective is to publish an anonymized version of the graph G, which still allows a broad class of queries to be answered accurately, but which maintains privacy of the associations. To make this goal precise, we describe our privacy goals, and outline classes of queries which we aim to answer.

Our privacy objective is based on the idea that in many cases it is the *association* between two nodes which is private and must be anonymized. As noted, the set of customers of a pharmacy may not be considered particularly sensitive, and the set of products which it sells may be considered public knowledge. However, the set of products bought by a particular customer is considered private, and should not be revealed. We focus on preserving the privacy of associations, and assume that properties solely of entities (e.g. state of a customer) are public. Clearly, there are situations with differing privacy requirements, commented on in Section 7.

Since it is desirable to allow answering of ad hoc aggregate queries over the data (e.g. how many customers from a particular zip code buy cold remedies), we wish to release some anonymized version of this data which gives accurate answers to such queries but protects the individual associations. More strongly, we want the graph properties of the data to be preserved. This corresponds to simple features, such as the degree distribution of the nodes, but also more long-distance properties, such as the distribution of nodes reachable within two steps, three steps, etc.

Here, as in all work on anonymization, there is an inherent tradeoff between *privacy* and *utility*, although this can be hard to quantify precisely. Various extreme approaches maximize one over another: publishing the original data unchanged clearly maximizes utility, but offers no privacy; removing all identifying information and publishing only an unlabeled (“fully censored”) graph gives high privacy, but limited utility for aggregate queries over nodes satisfying certain predicates.

Prior work has considered strong dynamic attack models (where nodes and edges can be inserted into the graph), which can result in some small number of associations being revealed [1]. For many situations we consider, this represents a very powerful attacker, and weaker attack models may suffice. It assumes that an attacker knows what data will be covered by the release and can easily modify it in advance. But, in the pharmacy example, adding edges means particular individuals must buy certain products in certain stores at certain times, which requires a very coordinated attacker. Adding nodes could involve creating new products for sale in the stores, which may not be plausible. Similarly, passive attacks require the attacker to collect complete and accurate information for a set of individuals. Even then, such attacks [1] only reveal information about entities for which some information is already known. Entities not involved in the attack remain secure. Clearly, there are cases where such attacks are possible and the results of [1] give a strong caveat; it is the responsibility of the data owners to determine against which attacks they should be secure.

In extreme cases, the unlabeled graph structure leaks information about individual edges: for example, if the underlying graph is complete then we know there is an edge between any pair from the censored graph structure alone. Or, if there are a few nodes with unique degrees and these degrees are known to the attacker, these nodes can be reidentified. But in typical cases such as the examples we consider, virtually nothing can be deduced from the graph structure alone. Again, the data owners must determine whether this level of disclosure is acceptable to them. Here we aim for privacy guarantees relative to the baseline of the unlabeled graph. In particular, we study what guarantees can be made in the following scenarios:

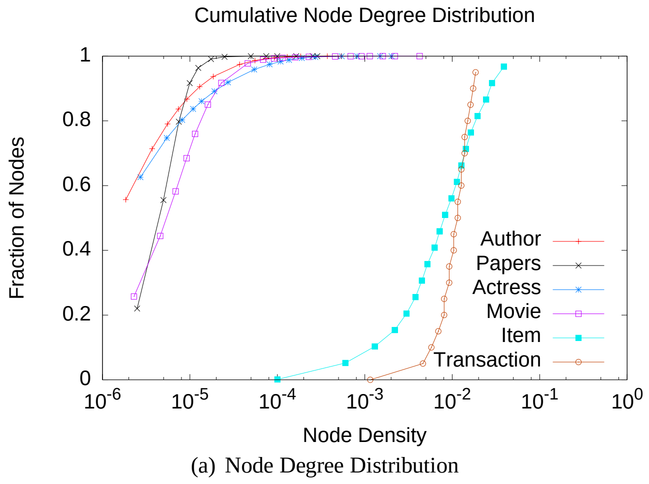
Definition 1 In the *static attack* case, the attacker analyzes solely the information which is published by the scheme, and tries to deduce explicit associations from this information. Ideally, the number of associations which can be correctly inferred (beyond what is implicit in the censored graph) should be minimal if not zero.

In the *learned link attack* case, the attacker may already know a few associations — for example, that customer c1 bought product p2 (as in Figure 1). The additional associations that can be inferred should be minimal if not zero when the number of link revelations is small.

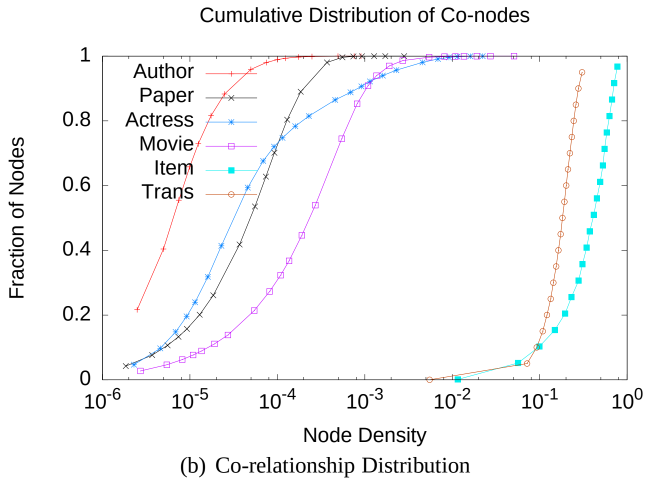
We are principally concerned with an attacker being able to make *positive* inferences, e.g. being able to deduce that c6 bought p6. We are less concerned about *negative* inferences, e.g. deducing that c1 did not buy p1. Since the graphs we analyze are sparse, and the maximum degree is at most a constant fraction of the total number of nodes, we consider such discovery to be entirely acceptable (the same assumption is implicitly present in much of the prior work on anonymizing tabular data, such as k-anonymity and permutation based methods). We are still concerned when some negative inferences eliminate enough possibilities to leave a positive inference: learning c4 bought at least one item but did not buy p1, p2, p3, p4, or p6 allows us to infer that c4 did buy p5. This form of positive inference is specifically captured and shown to be limited by our analysis.

2.3 Query Types and Utility

As in prior work, given the difficulty of giving a precise privacy/utility tradeoff, we consider approaches which first fix a given level of privacy and then try to optimize and measure the utility. In order to more precisely analyze utility,



(a) Node Degree Distribution



(b) Co-relationship Distribution

Fig. 3 Cumulative distributions of the degree and second-order degree of the three bipartite data sets

we describe a set of sample aggregate query types which we wish to support. We will measure the utility of our results by studying the accuracy with which these queries can be answered using the anonymized data. The queries can be based on predicates over solely graph properties of nodes (such as degree), which we denote P_n, and predicates over attributes of the entities, P_a. In our customer-product example, P_a could select out customers from NJ, or prescription products, while a typical P_n might be that a customer buys a single product. We separate these two types of predicates, since when we publish a censored graph, we can still evaluate P_n predicates exactly, while we have maximum uncertainty in applying P_a predicates.

We list a set of types of queries of increasing complexity, based on standard SQL aggregates (sum, count, avg, min, max):

– Type 0—Graph structure only: Compute an aggregate over all neighbors of nodes in V that satisfy some P_n.
  E.g.: Find the average number of products per customer; Compute the average number of customers buying only that product, per product.
– Type 1—Attribute predicate on one side only: Compute an aggregate for nodes in V satisfying P_a; Compute an aggregate on edges to nodes in V satisfying P_n from nodes in W satisfying P_a.
  E.g.: Find the average number of products for NJ customers; Find total number of CA customers buying only a single product.
– Type 2—Attribute predicate on both sides: Compute an aggregate for nodes in V satisfying P_a to nodes in W satisfying P'_a.
  E.g.: Count total number of OTC products bought by NJ customers; Total sales of Rx products to CA customers who buy nothing else.

Naturally, one can define yet higher orders of queries that are more complex, either through more constraints or more steps through the graph. Join-style queries would compute an aggregate of nodes from V at distance 2 from other nodes in V satisfying P_a connected via nodes in W satisfying P'_a, and so on. Other graph properties can be included in queries, such as measuring the diameter of an induced subgraph, or identifying particular patterns of interaction such as complete embedded subgraphs. For this work, we constrain our interest principally to the classes of queries defined above, since these are sufficiently rich to be challenging to answer accurately, while being sufficiently concise to specify compactly and work with over realistic data sets. In particular, note that while queries of type 0 can easily be answered on the fully censored graph exactly, answering queries of other types requires some more information about attributes of the entities in order to give any reasonable accuracy.

2.4 Datasets and Experimental Environment

All experiments for this paper were implemented in JDBC and SQL Server 2000. Experiments are performed on a variety of real and synthetic datasets representing bipartite graphs with quite distinct properties. The first dataset is from the Digital Bibliography and Library Project (DBLP), and consists of data about all conference papers collected by the project, and the authors of those papers. It was retrieved from `http://dblp.uni-trier.de/xml/` on 06/21/2007, and is available on request from the authors. The data set contains $|V| = 402023$ distinct authors, $|W| = 543065$ distinct papers, and $|E| = 1401349$ (author, paper) edges. The papers have additional attributes, such as year of publication, and the name of the conference, while the authors have attributes such as name (and, more generally, other attributes such as affiliation, although these are not represented within the DBLP).

The second dataset is from the Internet Movie Database (IMDB), and consists of data about all actresses referenced in the database, and the movies in which they are listed as appearing. It was retrieved from `http://www.imdb.com/interfaces` on 10/05/2007. The data set contains $|V| = 436727$ distinct actress, $|W| = 367874$ distinct movies, and $|E| = 1847630$ (actress, movie) edges. Here, the movies have attributes such as title and year. The two real data sets fit approximately the same schema, so we can translate queries on (author, paper) data into queries on (actress, movie) data.

A third data set is a synthetic transaction dataset from the Frequent Itemset Mining Dataset Repository (FIMI) web site. It can be accessed at `http://fimi.cs.helsinki.fi/data/T10I4D100K.dat`. We use the first 10,000 transactions in our simulation. In the FIMI data, there are $|W| = 866$ items involved in $|V| = 10,000$ transactions, and $|E| = 100550$ (transaction, item) pairs (edges in the graph). So it is an $\alpha = 0.01161$-sparse graph, and many items have very large degrees.

These datasets represent the kind of association we are interested in, with typical features of such graphs (a graph with varying sparsity, power-law degree distribution, non-random structure of links). Although the first two data sets have comparable numbers of nodes and edges, they display rather different graph properties, as illustrated in Figure 3. In Figure 3(a) we show the distribution of degrees of the various types of nodes (authors and papers from DBLP, actresses and movies in IMDB, items and transactions). Here, we represent the degree as the fraction of the largest possible degree, so an item which appears in 10 out of the 10,000 transactions has "density" 0.001. We show the cumulative distribution, so approximately 80% of the actresses in the IMDB have node density less than 10^{-5} (i.e. they are connected to at most a 10^{-5} fraction of all movies). Plotting the data in this way shows that the FIMI data, although smaller, is dramatically more dense.

The highest degree of an author is 290 (i.e. one author is associated with 290 conference papers), and the highest number of authors per paper is 115, while in both cases the total number of authors and papers is in the hundreds of thousands, making the density quite low. The most prolific actress has appeared in 744 movies, while the movie with the largest cast (actually a long running TV show listed as a single entity) has 1849 credited actresses. This indicates that these graphs are substantially sparse: although there are hundreds of thousands of nodes, the maximum degree of any node in DBLP is just a few hundred, and the maximum degree of a node in IMDB is less than two thousand, meaning that only a small fraction of the possible edges are present. Further, nodes of degree 1 are very common: many actresses are listed with only a single movie, many authors have only a single paper, and many movies credit only a single actress. Only for the papers is the number of single author papers exceeded by the number of dual author papers.

The second plot, Figure 3(b), shows the cumulative distribution of nodes reachable using two steps. This corresponds to, for example, the number of authors who are linked to a given author by a common paper; the number of movies linked to a given movie by a common actress; and so on. Here we see a clearer separation between the DBLP and IMDB data sets, indicating appreciably different structures. No author has more than 363 coauthors, while there is an actress who has been credited alongside a total of 9717 others. The reason for this is understandable: authors tend to choose their collaborations carefully, and may write several papers with the same coauthors. Meanwhile, actresses have less control over which other actresses they are cast with, and there is less tendency for particular pairings to be repeated. As a consequence, there is more local "clustering" within the DBLP data, in comparison to the IMDB data. We also see that the density of the FIMI data considering two steps is higher, and means that an appreciable fraction of the item nodes share a transaction in common with other nodes: about half the items share transactions with half of the other items, making this data set very dense indeed.

3 Applying Existing Techniques

A natural first approach to addressing these privacy questions is to apply prior work on table anonymization, since tables can represent graph data. However, such prior anonymization techniques only try to preserve the accuracy of table-based queries, and do not consider any graph semantics. As a result, we show that fundamental graph properties are quickly lost under such transformations, and we will see that even many of our type 0 queries are answered with intolerably high error. It is difficult to exhaustively try all existing methods, so we show that for three popular representative anonymization schemes the results are not usable over graph data.

3.1 Representing as a relation

Representing the customer-product example in Figure 1 using tables, gives a customer relation (Figure 1(a)), a product relation (Figure 1(b)), and a customer-product relation (Figure 1(c)). We can join these to make a single table (Figure 4(a)), and try to anonymize it. In our example, each row lists a customer, a product, the customer's state, and the product availability. How can we meet our goal of not revealing any (customer, product) association by applying a

Customer	Product	State	Availability
c1	p2	NJ	OTC
c1	p6	NJ	OTC
c2	p3	NC	OTC
c2	p4	NC	OTC
c3	p2	CA	OTC
c3	p4	CA	OTC
c4	p5	NJ	Rx
c5	p1	NC	Rx
c5	p5	NC	Rx
c6	p3	CA	OTC
c6	p6	CA	OTC

(a) Original data table

Customer	Product	State	Availability
*	*	*	OTC
*	*	*	OTC
*	*	*	OTC
*	*	*	OTC
*	*	CA	OTC
*	*	CA	OTC
*	*	*	Rx
*	*	*	Rx
*	*	*	Rx
*	*	CA	OTC
*	*	CA	OTC

(b) 3-anonymous data table

Fig. 4 Attempting to apply existing k anonymization to graph data

	p1	p2	p3	p4	p5	p6
c1	0	1	0	0	0	1
c2	0	0	1	1	0	0
c3	0	1	0	1	0	0
c4	0	0	0	0	1	0
c5	1	0	0	0	1	0
c6	0	0	1	0	0	1

(a) Matrix representation

	p1	p2	p3	p4	p5	p6
c1	*	*	0	*	*	*
c2	0	0	*	*	*	*
c3	*	*	0	*	*	*
c4	0	0	*	*	*	*
c5	*	*	0	*	*	*
c6	0	0	*	*	*	*

(b) 3-anonymized matrix

Fig. 5 Attempting to apply k-anonymization to data represented in adjacency matrix form

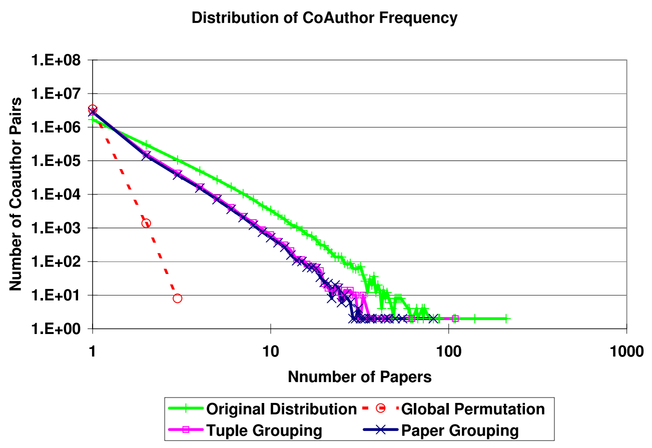
k-anonymization algorithm? Removing all customer IDs destroys all association structure from customers to products. Setting customer as a quasi-identifier and product as sensitive attribute fails because k-anonymization allows k products bought by the same customer to be grouped together (they share a quasi-identifier). Setting (customer, product) as the sensitive attribute fails, because k-anonymization does not alter or mask sensitive attributes. Instead, we could add a dummy sensitive attribute of "true" to each row to indicate that the association is sensitive. The k-anonymized version of this table must use generalization and suppression to ensure each row is indistinguishable from $k-1$ others [15,16]. Options for concealing customer and product identifiers are limited: since they are arbitrary identifiers, there is no natural hierarchy for generalization so they can only be withheld. A 3-anonymized version of our example data set shown in Figure 4(b) provides very low utility: for example, there is no natural way to obtain an accurate estimate of what fraction of customers bought only a single product.

This attempt at anonymization loses the notion of individual customers and products, and so is unable to give useful answers to the query types outlined above. Augmenting the anonymized data with some additional information risks breaching privacy and does not guarantee to anticipate all reasonable queries which could be formulated: recall that the purpose of publishing anonymized data is to allow a broad variety of ad hoc queries to be posed.

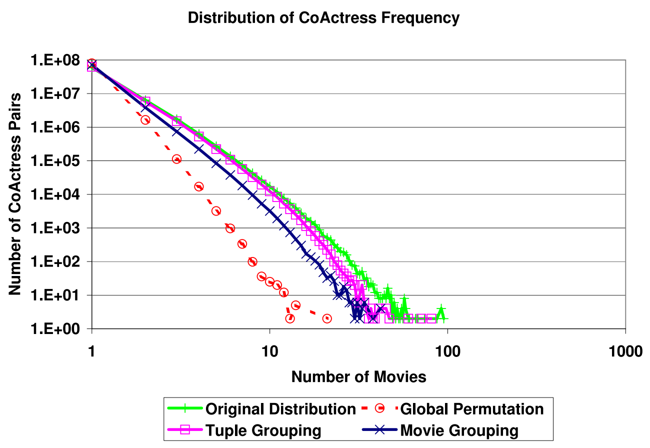
3.2 Representing as a matrix

A fundamental problem with the above approach is that k-anonymity is formally defined so that there should be at least k *individuals* whose representation is identical; in this representation, each individual is present in multiple places, so for example in Figure 4(b), two rows in the anonymized table refer to the same customer, giving them weaker privacy. This leads us to represent the graph data instead as a binary matrix: rows correspond to nodes in V, columns to nodes in W, and an entry (i, j) is set to 1 if there is an edge between $v_i \in V$ and $w_j \in W$, and 0 otherwise. We can now take such a matrix, and try to apply existing anonymization techniques on it. Similar to above, the only meaningful anonymization of a 0 or 1 value is to generalize to "*".

Applying k-anonymization is similar to having customer as a quasi-identifier and product as a sensitive attribute [15, 16]: now products with more than k buyers may be revealed, while unpopular products may be fully masked. This also virtually wipes out the utility of the data. For example, Figure 5(a) shows the matrix representation of the sample data from Figure 1, and Figure 5(b) shows the result of



(a) Co-authorship Frequency



(b) Co-Actress Frequency

Fig. 6 Effect of applying permutation methods on graph properties.

3-anonymizing it: only a few negative associations remain. The fundamental problem here is that these approaches have two equally unpalatable options: either an association is fully revealed, or else it is withheld.

3.3 Anonymization Through Permutation

The previous two approaches were based on blind application of tabular anonymization to graph data. A third approach to anonymizing tabular data is based on the idea of "permutation": breaking the links between quasi-identifier attributes and sensitive attributes [20, 23]. This seems more suited to the graph setting: we have an association between nodes in a graph that we wish to anonymize. This leads to the following algorithm: form edges into groups, and within each group, publish the pair of node (multi)sets that form edges. Grouping the edges from Figure 1 into sets of size 3 and 4 based on customer pairs gives:

$$(\{c1, c1, c2, c2\}, \{p2, p3, p4, p6\}),$$
$$(\{c3, c3, c4\}, \{p2, p4, p5\}),$$
$$(\{c5, c5, c6, c6\}, \{p1, p3, p5, p6\})$$

Equivalently, for a group containing edges $e_1 = (v_1, w_1), e_2 \dots e_\ell$ we generate ℓ permuted edges by picking a random permutation π and publishing $e'_1 = (v_1, w_{\pi(1)}) \dots e'_i = (v_i, w_{\pi(i)}) \dots e'_\ell$. Conceptually, imagine taking every edge in the group and "breaking it in the middle", then forming new edges by joining half-edges from V to half-edges from W. This method initially seems more promising than the above, since it guarantees to preserve node degrees (i.e. the number of products linked to a customer is the same before and after the permutation, and vice-versa), and the true mapping from customers to products seems well-masked. However, when we try to evaluate simple graph queries (type 0) over this data, we find that the results are highly inaccurate.

To see this in practice, we compared a variety of methods of generating permuted data, and evaluated some simple queries over the resulting data. The first method creates a "global permutation" of the DBLP and IMDB data where all edges are placed into a single group and permuted. A second method first creates smaller groups of nodes on each side of the bipartite graph, and then applies permutations within each group. For DBLP data, we formed groups by first sorting papers primarily by author count, conference and year. A "paper grouping" results from forming groups based on all edges relating to each consecutive pair of papers under this ordering. A "tuple grouping" results from forming groups based on each consecutive pair of edges under this ordering. For IMDB data, movies are first sorted by their actress count and year. A "movie grouping" and "tuple grouping" are found analogously to the DBLP case, by grouping all edges relating to pairs of movies, and to pairs of consecutive edges, respectively. Note that given the edge table, permutation of edges is the same as permuting the grouped ids, i.e., paper ids and movie ids, while keeping author ids or actress ids fixed.

Figure 6 shows the one hop neighborhood for each resulting permuted data set, over the original data set and three different permutations of it, on log-log scales. The results clearly demonstrate that permutation-based approaches do not accurately maintain the coauthor and coactress relationships for query answering. In particular, in the DBLP source data 1.6M pairs of coauthors have written at least one paper together, and Figure 6(a) shows that one pair has co-written 210 papers. In the global permutation, the maximum number of papers coauthored together is only three. The permutation of small groups via either tuple or paper grouping is closer to the source distribution, but the error is still significant: the frequency of coauthorship of a particular number of papers is underestimated by up to an order of magnitude. This is unsurprising since coauthors often collaborate over long periods, writing multiple papers together. Permutation of papers breaks this correlation and links unrelated authors.

In the IMDB data (Figure 6(b)), we observe the same trend, although not as pronounced as for the DBLP data. Tuple groupings tend to leave pairs of actresses linked through the same movie, but the movie grouping and global permutation clearly alter the distribution. This indicates that there is a higher correlation between pairings in the DBLP data than in IMDB. In the global permutation, the maximum number of movies in which a pair of actresses costar is 21, compared with 95 in the original distribution.

Other similar experiments based on different grouping criteria and different features of the distributions similarly failed to preserve these basic graph properties. Likewise, experiments based on grouping the other side of the graph (i.e. studying the co-paper and co-movie distributions) yielded equally low fidelity. Therefore we conclude that this permutation approach gives very poor answers to simple type-0 queries, and so is not suitable for further consideration, since we next propose a method which guarantees perfect answers to type-0 queries.

4 Privacy through Grouping

All the above attempts to use existing techniques render the data virtually unusable for the simple reason that they change or mask the graph structure in ways that fundamentally alter its properties. In contrast to the case of tabular data, where modifying a row has relatively minor impact on table properties, adding or deleting an edge can have significant impact on properties of a graph (for example, it can change a graph from being connected to disconnected). So we seek to avoid techniques which involve perturbing the graph structure. Instead, we focus on techniques which retain the entire graph structure but perturb the *mapping from entities to nodes*. That is, methods that publish a set of edges E' that are isomorphic to the original edges E, but where the mapping from E to E' is partially or fully masked. This technique is applicable in situations where it is considered safe to publish the unlabeled graph.

Outline. We define our grouping method in Section 4.1, and give a "safety" condition in Section 4.2 which ensures that privacy goals are met (proved in Section 4.3). We give a greedy algorithm to find a "safe grouping" (Section 4.4) and then show how to answer queries given the published anonymized grouping (Section 4.5). Lastly, we consider a special case where some groups are revealed exactly (Section 4.6), and provide experimental results (Section 4.7). Throughout, we introduce a variety of notation to represent various concepts within the grouping. A summary of the most important notation is presented in Table 1 for convenience of reference.

Table 1 Notation used in this paper

V	Set of nodes, with $\|V\| = m$
W	Set of nodes, with $\|W\| = n$
E	Set of edges from $V \times W$
P_a	Predicate on attributes of entities
P_n	Predicate on (graph) properties of nodes
(k, ℓ)-grouping	V split into size k groups, W into size ℓ groups
H	Function mapping nodes into groups
F_V, F_W	Functions renaming nodes given by Definition 3
R_V, R_W	Remapping functions given by Definition 3
$(k, \ell)^{*(q,r)}$	Modified grouping given by Definition 5
U, L, μ	Upper and lower bounds, and expected answer
U_j, L_j, μ_j	Upper, lower bounds & expected answer for group j
$U_{i,j}, L_{i,j}, \mu_{i,j}$	Upper, lower bounds and expected answer between groups i and j

4.1 Definition of Grouping

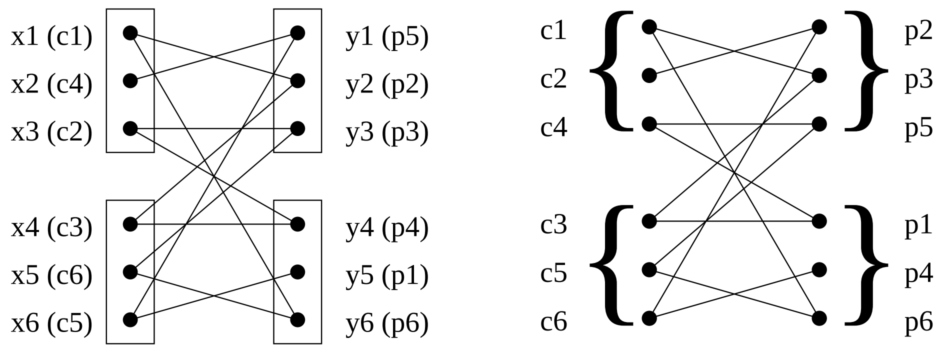
In this paper, we focus on masking the mapping via *grouping* the nodes of the graph. This technique preserves the underlying graph structure perfectly, but masks the exact mapping from entities to nodes, so for each node we know a *set* of possible entities that it corresponds to. The group size k is a parameter: larger k gives more privacy, but reduces the utility. We first provide formal definitions of groupings, illustrated by an example, and then show how these groupings enable the masking.

Definition 2 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,

$$\forall v \in V : \exists V_v \subseteq V : |V_v| \geq k \wedge (\forall v_i \in V_v : H(v_i) = H(v))$$

That is, the function H partitions V into subsets of size at least k. The k-grouping is *strict* if every group V_v has size *exactly* k or $k + 1$.

In other words, a k-grouping partitions V into non-intersecting subsets of size at least k. The strictness property insists all groups in a k-grouping be close to k in size, since smaller groups allow more accurate query answering. Given a set of nodes V, it is not always possible to divide them into groups of size exactly k, since $|V|$ may not be a multiple of k; however, it is always possible to divide them into groups so that all are size k or size $k + 1$, provided that $|V| \geq k^2$. Therefore, we describe such groupings as "strict", since they keep the group sizes as close to the parameter k as possible. As we add additional requirements to the grouping, we will see whether it is still possible to find strict groupings which



(a) Grouped and Relabeled Graph (b) Graph of Published Data

Fig. 7 Graphical Representation of grouped data

x1	y2
x1	y6
x2	y1
x3	y3
x3	y4
x4	y2
x4	y4
x5	y3
x5	y6
x6	y1
x6	y5

E'

Customer	Group
c1	CG1
c2	CG1
c3	CG2
c4	CG1
c5	CG2
c6	CG2

H_V

Product	Group
p1	PG2
p2	PG1
p3	PG1
p4	PG2
p5	PG1
p6	PG2

H_W

X-node	Group
x1	CG1
x2	CG1
x3	CG1
x4	CG2
x5	CG2
x6	CG2

R_V

Y-node	Group
y1	PG1
y2	PG1
y3	PG1
y4	PG2
y5	PG2
y6	PG2

R_W

Fig. 8 Published tables representing (3,3)-anonymization of example relation

satisfy these requirements. We next use the definition of grouping to publish a modified version of the graph:

Definition 3 Let F_V be a relabeling function to relabel elements of V injectively onto a new set $F_V(V)$; and let F_W be a relabeling function to relabel elements of W injectively onto a (disjoint) set $F_W(W)$. Given a k-grouping on V, H_V, and an ℓ-grouping on W, H_W, of a graph $G = (V, W, E)$, define the (k, ℓ)*-grouped graph* G' as $G' = (V, W, H_V, H_W, E', R_V, R_W)$ where:

(a) V and W are the original sets of entities V and W, and H_V and H_W are the grouping functions defined above.

(b) E' is the relabeled edge set given by

$$E' = \{(F_V(v), F_W(w)) | (v, w) \in E\}.$$

(c) R_V, R_W are remappings defined by

$$R_V(F_V(v \in V)) = H_V(v)$$
$$\text{and } R_W(F_W(w \in W)) = H_W(w).$$

When both H_V and H_W are strict, this is a strict (k, ℓ)-grouping.

Example 2 For the example in Figure 1, set groups CG1, CG2 (customer group 1 and 2) and PG1, PG2 (product group 1 and 2) as

$$H_C^{-1}(CG1) = \{c1, c2, c4\}$$
$$H_C^{-1}(CG2) = \{c3, c5, c6\}$$
$$H_P^{-1}(PG1) = \{p2, p3, p5\}$$
$$H_P^{-1}(PG2) = \{p1, p4, p6\}$$

where H^{-1} denotes the pre-image of its parameter under the function H.
This is a strict $(3, 3)$-grouping since every customer group and every product group has (exactly) three members. The resulting grouped graph is shown in Figure 7(a), with the arbitrary relabeling of nodes on xi's and yi's. The published information can be derived from this: Figure 8 shows the five published tables (in addition to the original customer and product tables, Figure 1(a) and 1(b)). The result is compactly represented as a graph in Figure 7(b): it shows the edge structure, and which sets of nodes map to which sets of entities, but hides the exact mapping. □

The grouping functions H_V and H_W contain most of the necessary information to specify the modified graph as a function of original graph G. This definition is well-suited to storage within a relational database. For example, in our customer-products example, we publish customer and product relations as before (corresponding to V and W); customer-group and product-group tables which encode the mapping of each customer and product to groups (corresponding to H_V and H_W); a masked-customer-product relation, in which each customer and product is mapped to a new node id (E'); and lastly masked-customer-group and masked-product-group tables which map from the masked identifiers to groups (R_V and R_W). Note that the base relations corresponding to V and W should not contain any information relating to the graph, such as the degree of the node. Otherwise, an attacker could potentially use this to re-link between rows of V or W and nodes in E'. Two further examples of groupings illustrate extremes of the privacy-utility tradeoff:

Example 3 **Smallest groups.** Setting $H_V(v) = v$ and $H_W(w) = w$, (the identity functions) gives a $(1,1)$-grouped graph G'. Here, $E' = E$, and hence G' encodes the original graph G exactly. Every query on G' can be answered with the same accuracy as on G. So there is perfect utility, but no more privacy than we began with. □

Example 4 **Largest groups.** Setting H_V to map all $m = |V|$ members of V to the same group, say 0, and H_W to map all $n = |W|$ members of W to, say, group 1 gives the (m,n)-grouped graph G'. G' has no useful information mapping between entities in V, W and the nodeset of E'. That is, we publish entity tables and the fully censored graph. Recall that we are assuming it is acceptable to publish a censored graph, and so we say that this grouping guarantees the same level of privacy. This is the case where the mapping from entities to nodes is completely hidden. In the customer-product example, this entails publishing the customers relation and products relation unchanged (since these are not considered private). In addition, we apply an injective masking function F on the customer-product relation so that each (c,p) pair is mapped to $(F(c), F(p))$, and publish the resulting censored table. This retains the graph structure, as required, but completely removes the mapping from entities (e.g. customers and products) to nodes in the graph. We cannot have any more privacy in our setting, when we insist on publishing at least this much information. This offers very limited utility in answering query types 1 and 2 listed in Section 2.3, since we cannot apply any selective attribute predicate with any certainty. □

Privacy-Utility Tradeoffs. Between these two extremes lie many possibilities that trade off utility and privacy. Given a (k,ℓ)-grouped graph, where both k and ℓ are fairly small, aggregate queries such as those described in Section 2.3 can be answered approximately. Bounds can be placed on the answers within which the true answer must fall (Section 4.5). When k and ℓ are small, these bounds are narrow; as k and ℓ grow large, the bounds will widen accordingly. Clearly, a (k,ℓ)-grouping offers more utility (and less privacy) than a (k',ℓ)-grouping if $k < k'$; the same holds true between (k,ℓ)- and (k,ℓ') groupings for $\ell < \ell'$. But we cannot easily compare (k,ℓ)- and (k',ℓ')-groupings unless $k < k'$ *and* $\ell < \ell'$. Thus, choices of k and ℓ define a *lattice* over possible groupings, bounded by $(1,1)$ and (m,n). We explore several points in this space in more detail; Figure 9 shows the lattice structure, including points of note that are defined and discussed in subsequent sections. We will investigate these points in greater detail in subsequent sections, as we analyze how to choose groupings in order to give privacy guarantees, and how to effectively answer aggregate queries on grouped graphs.

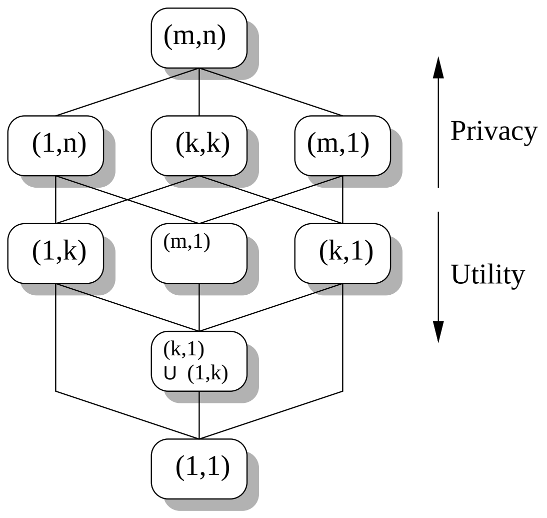


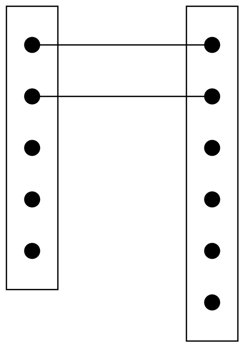
Fig. 9 Lattice over groupings and privacy/utility tradeoff

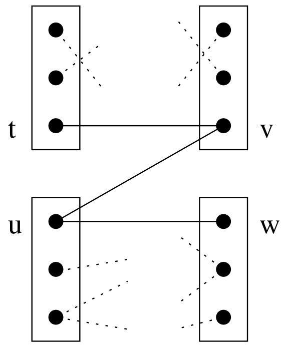
4.2 Safe Groupings

There are many ways to form a k-grouping, but not all of these offer the same level of privacy, due to the local graph structure. We introduce the condition of "safety" which ensures privacy holds even under revelation of certain information.

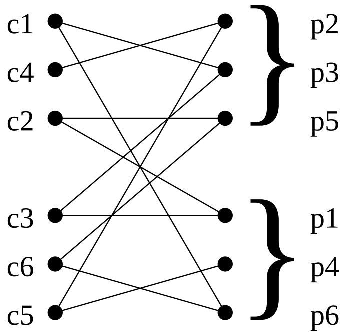
Example 5 Consider a large graph G, which happens to contain the complete subgraph between nodes $\{v_1, v_2, v_3\}$ and $\{w_1, w_2, w_3\}$. Suppose we form 3-groupings on V and W so that $\{v_1, v_2, v_3\}$ forms the entirety of one group in H_V, and $\{w_1, w_2, w_3\}$ forms the entirety of a group in H_W. From the published G', it is possible to infer immediately all the connections between these six nodes (a static attack). Such inference is not possible on the fully censored version of G', but the unfortunate choice of grouping allows information to leak. □

Essentially, this is a problem of lack of *diversity*: since the interaction pattern between the two groups is too uni-


(a) Structure of safe groups



(b) (t, v) and (u, w) imply (u, v)



(c) Safe $(1, 3)$-grouped graph

Fig. 10 Safe Grouping Algorithm and Examples

formly dense, such undesired inference is possible. Some natural attempts to fix this, such as insisting that the density of edges between any pair of groups is low, are not guaranteed to still hold as edges are learned by an adversary. We define a stronger notion of "safe grouping", which we subsequently prove is robust against static and learned link attacks.

Definition 4 H_V is a *safe* grouping of V in the context of a graph $G = (V, W, E)$, if the following condition holds:

$$\forall v_i \neq v_j \in V : H_V(v_i) = H_V(v_j) \Rightarrow \\ \nexists w \in W : (v_i, w) \in E \wedge (v_j, w) \in E$$

By extension, a (k, ℓ)-grouping of a graph G is safe if H_V and H_W are both safe groupings.

That is, a safe grouping ensures that any two nodes in the same group of V have no common neighbors in W (the definition for a safe grouping of W is symmetric, interchanging the roles of V and W). This ensures a level of sparsity between groups, but goes further in restricting the pattern of allowed links. In the customer-products example, it means that no two customers in the same group have bought the same product if the grouping is safe. Hence, the groupings in Figure 8 are safe. Given G and $k > 1$, there is no guarantee that there exists a safe k-grouping (all 1-groupings are trivially safe), but in practice they are easy to find (Section 4.4).

A necessary condition for the existence of a safe (k, ℓ)-grouping arises from the *sparsity* of the graph. A group of size k in V and a group of size ℓ together induce a subgraph of G which could have at most $k\ell$ edges. However, if the grouping is safe then (within the induced subgraph) any node can have degree at most 1; otherwise, there are two nodes with a common neighbor. Figure 10(a) shows a typical structure between two groups of size $k = 5$ and $\ell = 6$. So there can be at most $\min(k, \ell)$ edges between these two groups. This is true for every possible pair of groups. Since every edge touches exactly two groups, the α-sparsity of the subgraph, defined by $\alpha = |E|/|V||W|$, can be at most $\min(k, \ell)/(k\ell) = 1/\max(k, \ell)$.

We next show that finding a k-grouping when all groups are forced to be size k can be hard even for small values of k:

Theorem 1 *Finding a safe, strict 3-grouping is NP-hard.*

Proof Define $G^2(V) = (V, E^2)$ as the graph on V so that

$$(v_i, v_j) \in E^2 \iff \nexists w \in W : (v_i, w) \in E \wedge (v_j, w) \in E.$$

The requirement on H_V to be safe is equivalent to requiring that every pair of nodes in the same group must form an edge in E^2. That is, the group of nodes in the grouping forms a clique in (non-bipartite) $G^2(V)$. Therefore, a strict 3-grouping of V corresponds to a partition of $G^2(V)$ into triangles (forcing each group to be size 3). For any desired graph $G_1 = (V_1, E_1)$, define a bigraph $G = (V, W, E)$ such that $G^2(V) = G_1$: create $V = V_1$ and $W \subseteq V \times V$, and for each $(v_i, v_j) \in E_1$, insert $(v_i, (v_i, v_j))$ and $(v_j, (v_i, v_j))$ into E. Since partitioning a graph into triangles is NP-hard (problem [GT11] in [5]), and we can encode this problem as an instance of finding a safe, strict 3-grouping, we conclude that this problem is NP-hard also. $\square$

However, safe groupings can be found easily when the graph is sparse enough. For a bigraph $G = (V, W, E)$ where every node has degree 1 (i.e. E gives a matching between V and W), every possible grouping is safe, trivially. More generally, when the graph is sparse and does not have nodes which have (almost) all possible neighbors, safe k-groupings can be found for practical values of k (10^0 – 10^2, say). Intuitively, the constraints posed by the edges of the graph are easy to satisfy when not too many edges are present.

Most of the graph types discussed already are quite sparse and have few nodes of high degree: most shoppers purchase only a small number of the items on sale in a store, and most items are purchased by a fraction of all shoppers; most authors write only a small number of papers relative to the total number of papers written, and most papers have a small number of authors. Studying the data from

DBLP, we observe that the most prolific author has written 290 papers (out of 500K), and the most authors on a single paper is 115 (out of 400K). In total, there are only 1.4M edges in the author-paper graph, out of a possible $400K \times 500K = 200,000M$, demonstrating that typical association data is very sparse ($\alpha = 7 \times 10^{-7}$-sparse, and $\beta = 0.00058$-sparse, as defined above).

4.3 Security of (k, ℓ)-Groupings

Figure 8 shows a $(3, 3)$ grouping. We analyze what can be deduced by an attacker presented with a safe (k, ℓ)-grouping of graph data, where at least one of k and ℓ are greater than 1. We first argue that safe groupings are secure against the static attacks defined in Definition 1. Our arguments work by enumerating all the possible configurations, or *possible worlds* that are consistent with the data that is published.

Lemma 1 *In a safe grouping, given nodes $v \in V$ and $w \in W$ in groups of size k and ℓ respectively, there are $k\ell$ possible identifications of entities with nodes and the edge (v, w) is in at most a $1/\max(k, \ell)$ fraction of such possible identifications.*

Proof Consider a group V_G of V containing k nodes, and a group W_G of W containing ℓ nodes. In the subgraph of G induced by V_G and W_G, there are $e \leq \min(k, \ell)$ edges, following from the definition of safe grouping. There is no information available in what is published to break the symmetry between the nodes of V_G, or between the nodes of W_G. Hence, there is no published information which allows an observer to distinguish among the nodes, and so no way to prove that v and w are linked. Recall, we insist that tables V and W contain no data related to the graph itself, such as degree or neighborhood, that could break this symmetry.

For any entities $v \in V_G$ and $w \in W_G$, it is feasible that (v, w) is an edge, and also feasible that (v, w) is not an edge. More strongly, consider the number of ways of identifying entities v and w with the anonymized nodes $\{x_1 \ldots x_k\}$ and $\{y_1 \ldots y_\ell\}$. Since all $k\ell$ possibilities are feasible, then there is an edge between v and w in exactly an $e/k\ell$ fraction of feasible configurations (possible worlds), i.e. at most $\min(k, \ell)/k\ell = 1/\max(k, \ell)$, the bound on the density of the whole graph derived in Section 4.2. Since this analysis holds for every pair of groups, then the (static) attacker cannot infer any associations with certainty. □

Certainly, an attacker viewing the data can deduce the set of possible worlds that are consistent with the published data being an anonymization of that possible world. Adopting a probabilistic view, if the attacker has no prior beliefs about the entities involved, their beliefs mean that each possible world can be considered equally likely. By the above analysis, their probability of correctly guessing that an edge (v, w) was present in the original data is bounded by $1/\max(k, \ell)$.

Under this measure, a $(k, 1)$-grouping offers the same static guarantee as a (k, k)-grouping. However, as we discuss in more detail in Section 4.6, there are other factors to consider. We remark on a connection to the concept of ℓ-diversity [11]: here, the requirement is that between two groups the fraction of sensitive information (associations that are present) is bounded by $1/\max(k, \ell)$, which is similar to the ℓ-diversity requirement (it is also similar to other measures proposed for maintaining privacy in tabular data, such as the m-invariance requirement in re-publishing data [21]). If there are small groups, the attacker's confidence in a particular association can be higher. In particular, two groups of size 1 with an edge between them corresponds to a known association between entities. Although a safe (k, ℓ)-grouping has no groups of size 1, in the active (learned link) attack model, when an attacker learns the existence of an edge (v, w), he may be able to refine the grouping in order to create groups of size 1. We will show that this refinement has bounded impact on the security of entities not directly impacted by the edge revelation, after presenting an example where an attacker may learn an association.

Example 6 Consider the four groups shown in Figure 10(b), and the three edges that connect them. Other nodes in the same groups have edges to other groups (dashed lines) which do not affect this example. In the static case, as proved above, the attacker cannot make any strong inferences. However, in the link learning case, if the attacker learns (t, v) is an edge, he can use the fact that there is only one edge between the group of t and the group of v to identify t and v with nodes in the anonymized graph. Likewise, learning (u, w) allows u and w to be identified with the nodes that represent them. As a consequence, the attacker can infer that (u, v) is an edge, no matter how many other nodes are in the groups. □

The example shows that revealing an edge may allow an attacker to learn more about the nodes that it connects, and so infer more about the connections between such nodes. But the amount revealed about entities for which the attacker does *not* have information is minimal. A relaxed grouping definition allowing a few groups of size one enables this intuition to be formalized.

Definition 5 Define a $(k, \ell)^{*(p,q)}$-grouping as 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.

Observe that a (k, ℓ)-grouping is also a $(k, \ell)^{*(0,0)}$-grouping. Also, by applying Lemma 1, we note that a safe $(k, \ell)^{*(p,q)}$-grouping still gives a lot of privacy for nodes in the grouping: between a group of size k and one of size ℓ, each possible edge is present in at most a $1/\max(k,\ell)$ fraction of possible configurations, as before. But also, between a group of size 1 and one of size ℓ, there can be at most one edge in a safe grouping, and (also by Lemma 1) the edge is present in at most a $1/\ell$ fraction of possible configurations. Symmetrically, between a group of size k and one of size 1, the (at most one) edge is present in at most a $1/k$ fraction of possible configurations. Only between two groups of size one can we infer the existence (or absence) of an edge with certainty. As before, this fraction of possible configurations translates into a probability of correctly guessing an edge, under an appropriate probabilistic interpretation.

Theorem 2 *In the learned link case, given a safe (k,ℓ)-grouped graph and $r < \min(k,\ell)$ true edges, the most an attacker can infer corresponds to a $(k-r,\ell-r)^{*(r,r)}$-grouped graph.*

Proof This is shown by induction over the revelation of r edges. The base case $r = 0$ yields the $(k,\ell)^{*(0,0)}$-grouped graph. In the inductive case, there is a $(k-r,\ell-r)^{*(r,r)}$-grouped graph, and an additional edge (v,w) is learnt. As shown in the example above, in the worst case, this is enough to identify which node in the anonymized graph is v and which is w. This corresponds to refining of the groups: if v was in a group of size at least $k-r$, it is effectively split into a group of size 1 (containing v alone), and the remaining nodes now form a group of size at least $k-r-1$. Likewise, the group containing w is split into one of size 1 containing w alone, and one of size at least $\ell-r-1$. The resulting grouping is therefore at least a $(k-r-1,\ell-r-1)^{*(r+1,r+1)}$-grouping.

Observe however, that the identification of v and w reveals nothing about any other nodes, even those connected to v and w. More precisely, the resulting grouping is still *safe* by Definition 4. The crucial observation is that any refinement of a safe grouping by partitioning groups into smaller pieces remains safe. By appealing to Lemma 1, the attacker cannot infer any associations beyond those that are revealed by the grouping directly (i.e. only those links between groups of size one). This is sufficient to bound the new knowledge by the $(k-r,\ell-r)^{*(r,r)}$-grouping. □

This is directly comparable to results on tabular data k-anonymization where the aim is to ensure that individuals are secure up to the revelation of $k-1$ pieces of information about other individuals. Here, individuals and their associations are secure up to the revelation of $k-1$ pieces of information (edges) about others.

4.4 Finding a safe grouping

We describe a greedy algorithm to find a safe k-grouping of V. Precomputing the self-join of the edge table E on W allows quickly testing whether it is safe to put two nodes in the same group. For each node u in turn, the algorithm attempts to place u in the first group of the partial grouping with fewer than k nodes. If this would make the grouping unsafe, it tries the next group, and so on. If there is no group that meets these requirements, then a new group is started, containing u alone. After processing all nodes, there may be some (few) groups with fewer than k nodes in them. The algorithm collects these nodes together, and reruns the above loop allowing for groups of size $k+1$ instead of k. If the graph is sufficiently sparse, then a safe grouping in which every group has either k or $k+1$ nodes in is produced, and so the grouping is strict. Else, the algorithm continues but now allows groups up to size $k+2$, and so on. Eventually, either a safe grouping is found, or the algorithm terminates once some large group size is reached. In this case, the method fails, but can be run again by choosing a different ordering of the nodes, or by picking a smaller value of k.

```
Algorithm 4.1: GROUP($V, W, E, k$)
 $j \leftarrow k$;
 $\forall i : VG_{i,j} \leftarrow \emptyset$;
 repeat
   for $u \in V$
     do
       $i \leftarrow 1$;
       while ($\exists v \in VG_{i,j}, w \in W : (v,w) \in E$
           $\wedge (u,w) \in E) \vee |VG_{i,j}| > j$
         do $i \leftarrow i + 1$;
       $VG_{i,j} \leftarrow VG_{i,j} \cup u$;
   $j \leftarrow j + 1$;
   $V \leftarrow \emptyset$;
   $i \leftarrow 1$;
   $l \leftarrow 1$;
   for $i : (|VG_{i,(j-1)}| > 0)$
     do
       if $|VG_{i,(j-1)}| \geq k$
         then $VG_{l,j} \leftarrow VG_{i,(j-1)}$;
              $l \leftarrow l + 1$;
         else $V \leftarrow V \cup VG_{i,(j-1)}$;
 until $|V| = 0$
```

Fig. 11 Pseudocode to find safe k-grouping

Pseudo-code of this heuristic is shown in Figure 11. Initially, all groups in the jth iteration ($VG_{i,j}$) are set to empty. Then for each node u in the current set of nodes not allocated to groups, V, the algorithm tries each group in turn; if adding the node to that group would violate the safety condition, or cause the size of the group to exceed the current size limit j, then it moves to the next group (eventually, it will find an empty group, in which the item can be placed safely). Once all nodes have been processed, the algorithm then iterates over each non-empty group. If the group is not too small, then it gets copied as a group for the next iteration; else, the group is too small, and the nodes are returned to the set V for processing in the next round. The process terminates when all groups are at least size k, and by their formation must constitute a safe grouping.

In our experiments this heuristic easily found strict safe k-groupings for small values of k. There is the opportunity to optimize by choosing an initial ordering for the nodes, with the aim of giving better accuracy on queries. When a selective predicate is evaluated over a group, tighter query bounds are given when either (almost) all nodes in the group are selected, or none are selected. When a handful of nodes are selected from a group, there will be more uncertainty in answering the query. Putting similar nodes in a group together will therefore give higher accuracy. It is tempting to do this based on attributes of the entities. However, this can permit attacks in the style of the minimality attack defined in [19]: knowing that groups were formed in a particular way allows an attacker to deduce the identity of nodes, and hence infer associations.

Instead, if groups are chosen solely on graph properties, then we can publish the grouping algorithm, and anyone will find the same groups of nodes given the same unlabeled graph, so no information relating to the mapping of nodes to entities derives from the choice of which nodes to group together. This still gives many possibilities. For example, to improve accuracy on queries involving graph properties such as node degree (e.g. selecting customers buying a single product), sorting by node degree will greatly improve query answering. The sorted list of degrees of neighbors can break ties. Other arrangements are possible; in our experimental evaluation we will compare the groupings found by an arbitrary ordering of the nodes to one based on first sorting in the manner outlined.

4.5 Query answering on (k, ℓ)-grouped graph

We show that aggregate queries of the type considered in Section 2.3 can be answered accurately and efficiently from a published (k, ℓ)-grouped graph. First, since E' is isomorphic to E and queries of type 0 are solely on the underlying graph structure, they can be answered exactly. Queries of type 1 and 2 cannot guarantee perfect accuracy, since it is not possible to determine exactly which nodes their predicates select. However, they can be answered approximately, by providing bounds and expected values on the aggregate query. The core of our approach to query answering is to consider the set of configurations that are consistent with the published data: that is, from the anonymized data, consider all possible inputs (or "possible worlds") which could have resulted in this anonymization being produced. The result of the grouping compactly encodes this set of possible worlds. Typically, there are exponentially many possible worlds compared to the size of the published data, but by carefully using the structure of the anonymized data it is possible to extract bounds on the answer to aggregate queries, and expected values. We will also show that obtaining the tightest bounds on the query answer is NP hard, and so there is little prospect for doing better than materializing every possible world. We later show empirically that the weaker bounds obtained are quite usable in practice. It is beyond the scope of this paper to cover all possible forms of aggregate queries that could be posed, so we instead analyze various typical cases that illustrate the main ideas.

A typical type 2 query is of the form "count the total number of OTC products bought in NJ". Since the set of products within each group is known, the number of nodes selected by the product predicate in a group is easily found. The same is true for any customer group. The tightest bounds follow from evaluating the query over all possible assignments of entities to nodes, but this would be very costly, as the following theorem argues:

Theorem 3 *Finding the best upper and lower bounds for answering an aggregate query of type 2 is NP-Hard.*

Proof The hardness of the tight upper bound problem is shown by a reduction from the set covering problem [5]. Given subsets $S_1, \ldots, S_t$, whose union is $U = \{a_1, \ldots, a_u\}$, construct a bipartite graph (V, W, E). For each subset S_i, create a node v_i in V. All nodes in V are placed into a single group of size t. For each $a_i \in S_j$, create a node w_{ij} in W, and an edge (v_j, w_{ij}). W is partitioned into groups corresponding to the same a_i, i.e., group $G_i = \cup_j \{w_{i,j}\}$. The grouping of the graph is safe, by construction. To decide whether there exists k subsets that cover U, we set our problem as follows: the query selects k nodes in V, and exactly one node from each group of W. There is a set cover of size k if and only if the answer to the tight upper bound problem is $|U|$.

The hardness of the tight lower bound problem is shown by a reduction from the maximum independent set problem [5]. Given an undirected graph $G_1 = (V_1, E_1)$, construct a bipartite graph $G' = (V, W, E')$ similarly to the proof of Theorem 1: for each edge $(v_i, v_j) \in E_1$, insert $(v_i, (v_i, v_j))$ and $(v_j, (v_j, v_i))$ into E', and create a group of size 2 containing the two nodes (v_j, v_i) and (v_i, v_j). All nodes in V are put in a single group. Again, the grouping of G' is safe by construction. To decide if there exists an independent set of size k in G, set the query to select k nodes in V, and only one node in each group of W. There is an independent set of size k if and only if the tight lower bound for this query is 0. $\square$

Instead, slightly weaker bounds are obtained by considering each pair of groups in turn to find bounds on the query answer. The answers can then be combined to give the overall bounds. This approach certainly gives correct lower and upper bounds, but may be loose: one part of the bound may derive from one assignment of nodes to entities within a group, while another part may result from a distinct assignment within the same group. So the resulting bound is not compatible with any realisable configuration, and so may be loose.

Example 7 Consider answering the query "Count the total number of OTC products bought in NJ". We analyze each pair of groups in turn. Given a safe group CG_i of k_i customers, of whom a_i are NJ customers; a safe group PG_j of ℓ_j products, of which b_j are OTC products; and c_{ij} edges between the two groups, we can find the following bounds:
(i) Upper bound. Between the pair of groups, there can be a contribution of at most $U_{i,j} = \min(a_i, b_j, c_{ij})$ to the query. For a given product group PG_j, the total contribution over all customer groups CG_i to the query is no more than $U_j = \min(\sum_i U_{i,j}, b_j)$. Summing this over all product groups gives an upper bound of $U = \sum_j U_j$.
(ii) Lower bound. For a given pair of customer group CG_i, and product group PG_j, there is a contribution of no less than $L_{i,j} = \max(0, a_i + b_j + c_{ij} - k_i - \ell_j)$ to the query. For a given product group PG_j, the bound over all customer groups is $L_j = \max_i L_{i,j}$. We can sum this to get an overall lower bound, $L = \sum_i L_i$.
(iii) Expected answer. With no other information than what is published, the best strategy is to treat all assignments of nodes to entities as equally likely. Under this assumption, the expected selectivity between CG_i and PG_i from the product perspective is $\mu_{ij} = \frac{a_i b_j c_{ij}}{k_i \ell_j^2}$. Over all customer groups, the estimated Expected Bound for the jth product group is $\mu_j = \ell_j(1 - \prod_i(1 - \mu_{i,j}))$, assuming independence between the groups and using the inclusion-exclusion principle. It can be argued that that this approach is well-founded, since all possible assigments of nodes to entities are possible. The expected answer for the query is then $\mu = \sum_j \mu_j$. These can be verified by simple case analysis over the structure in Figure 10(a). □

Such queries can be answered in time $O(|E|)$, since each edge in the original graph connects a single pair of groups, and for groups with no edges between them ($c_{ij} = 0$), $U_{i,j} = L_{i,j} = \mu_{i,j} = 0$.

Example 8 The query "Find the maximum number of CA customers buying a single Rx product" can be answered by considering in turn each node that could possibly be a CA customer (is in a group which contains at least $a_i \geq 1$ CA customers), and finding exactly the products bought alone associated with that node. Upper and lower bounds increase if there are $b_j \geq 1$ Rx products or no fewer than ℓ_j Rx products in the product's group of size ℓ_j, respectively. These imply upper and lower bounds on the global maximum. Similarly, expected bounds follow by assuming a customer has probability of being in CA with probability a_i/k_i in a group of k_i customers; and that a product in a group of ℓ_j product has probability b_j/ℓ_j of being prescription only. □

As above, since we have to do a constant amount of work for each edge in the original bigraph, the computational cost is $O(|E|)$.

4.6 $(k, 1)$- and $(1, \ell)$-Groupings

A significant class of groupings arise when all groups of one set of nodes are of size 1. These are $(k, 1)$- (or symmetrically, $(1, \ell)$-) groupings. Here, more is revealed about associations between entities of the *same* type (our focus up to now has been on associations between entities of *differing* types), since the true mapping from one set of nodes to entities is revealed. In the customer-products example, a $(1, \ell)$-grouping reveals exactly how many products a particular customer has bought, who has bought the same product, etc., while still protecting the exact associations.

Example 9 Figure 10(c) shows a safe $(1, 3)$-grouping of our example data. The corresponding published tables are H_W and R_W as shown in Figure 8; H_V and R_V are not needed, since V maps directly onto the nodes of E'. Despite this information being revealed, the private associations between customers and products are still hidden: although Figure 10(c) shows that customers c1 and c3 bought the same product, it could be any one of $\{p2, p3, p5\}$. □

This again resembles a diversity requirement similar to ℓ-diversity: any customer is known to have bought one product out of a group of ℓ. From Lemma 1 and Theorem 2, given a safe $(k, 1)$-grouping, any edge still is between one of k equally likely nodes of V, and given r edge revelations, an attacker is still faced with a $(k-r, 1)^{*(r,0)}$-grouped graph. While information is revealed about interactions between one set of nodes (customers, in the example above), in many cases, this information release may be permissible. Our above results show that there are still strong guarantees on the privacy of associations, while revealing information such as, from pharmacy sales, which medicines were bought by the same person (without revealing who that person is). If it is acceptable to release such information, some queries are answered with higher accuracy.

Query answering on $(1, \ell)$ and $(k, 1)$-grouped data. Queries are answered in much the same way as in the more general (k, ℓ) case. However, many queries are answered more accurately, since the amount of uncertainty is reduced, and the anonymization entails fewer, more consistent possible worlds. The impact on our bounds is that in Examples 7 and 8, $a_i = k_i = 1$ or $b_j = \ell_j = 1$, simplifying the bounds. In particular, some queries of type 1 can be answered exactly: if the predicate is on the 1-grouping, the correct set of entities can be found exactly, which allows the exact answer to the aggregate query to be found. Type-2 queries can be answered with tighter bounds:

Example 10 For the query of Example 7 over a $(k,1)$-grouped graph, the set of OTC products is known precisely. For each OTC product, we add 1 to the upper bound if they have a buyer in a group which contains an NJ customer; and add 1 to the lower bound if they have a buyer in a group in which everyone is in NJ. For the expected bound, the expectation that a customer in a group of size k_i with a_i NJ customers is $\mu_{i,j} = a_i/k_i$, so the probability of any buyer of the product being from NJ is $1 - \prod_i (1 - E\mu_{i,j})$. Similarly, for Example 8, we can consider all single products bought by NJ customers exactly, and find the corresponding bounds (upper, lower, and expected) on which are prescription only. □

4.7 Experimental Analysis of Utility

In this section, we evaluate the utility of the anonymized data through experiments on the DBLP and IMDB data. Specifically, we study the accuracy of three sample queries with different properties. For each query, we compute a lower bound estimation L, an upper bound estimation U, and an 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 $\frac{|\mu-Q|}{Q}$. To clearly show the trends, we repeat each experiment over ten random choices of predicates and show the mean error bounds. We describe the experimental setup firstly in terms of the DBLP data; since the datasets have essentially the same schema structure, the equivalent queries on the IMDB data are formed by replacing authors with actresses, and papers with movies. The three queries are:

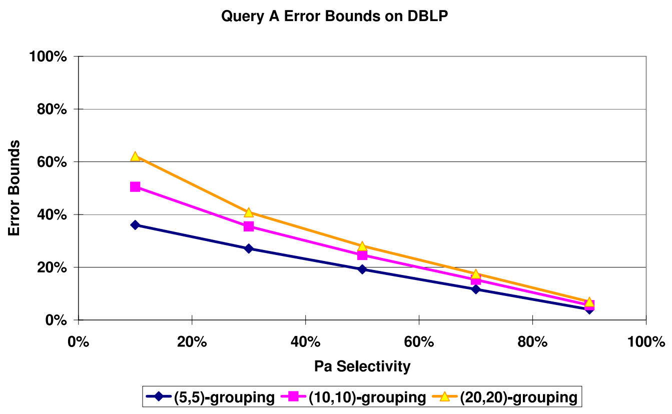
- **Query A:** *Find the average number of authors of any paper satisfying predicate P_a (equivalently, find the average number of actresses in any movie satisfying the predicate)*. This is a type-1 query with an attribute predicate only. We vary the selectivity of P_a from 10% to 90%.
- **Query B:** *Find the total number of single author papers satisfying P_a (single actress movies satisfying P_a).* This is also a type-1 query with both attribute predicates and structural predicates. The selectivity of P_a is varied as above, while the single author predicate is kept constant. We can make use of the bounds derived in Section 4.5. For each group, the query predicates will select a matching entities (authors or actresses), and c matching edges (edges incident on nodes with degree 1). We obtain upper and lower bounds per group j of size k as $L_j = \max(0, a+c-k)$, and $U_j = \min(a,c)$. Then we sum over all groups to get the final bounds L and U. For the expected answer, for each paper (movie) in a group of size k, the probability that it is selected is $\frac{a}{k}\frac{c}{k}$. Thus for each group j, the expected number of matching entities is $\mu_j = \frac{a \cdot c}{k}$. Then we can sum all groups and get the final expected value μ.
- **Query C:** *Find the total number of papers satisfying P_a having authors who satisfy P'_a (movies satisfying P_a starring actresses who satisfy P'_a).* This is a type-2 query, which is answered again by applying the methods of Section 4.5. We vary the selectivity of both P_a and P'_a.

These fit exactly the form of the queries we have studied in Example 7 and Example 8 (note that type-1 queries can be thought of as type-2 queries where one of the attribute predicates is always true). We do not consider any type-0 queries, since our earlier analysis shows that they can be answered exactly from the graph structure alone. We computed groupings over the papers and authors in the DBLP data described in Section 2 using the method detailed in Section 4.4. We built 20-groupings, 10-groupings, and 5-groupings over the data. The first iteration of the algorithm was able to find safe k-groupings covering almost every node: the 20-grouping of papers had 43 papers (out of 540K) not in groups of size 20, while there were just 3 authors not in groups of size 20. The next iteration easily found a safe, strict 20-grouping.

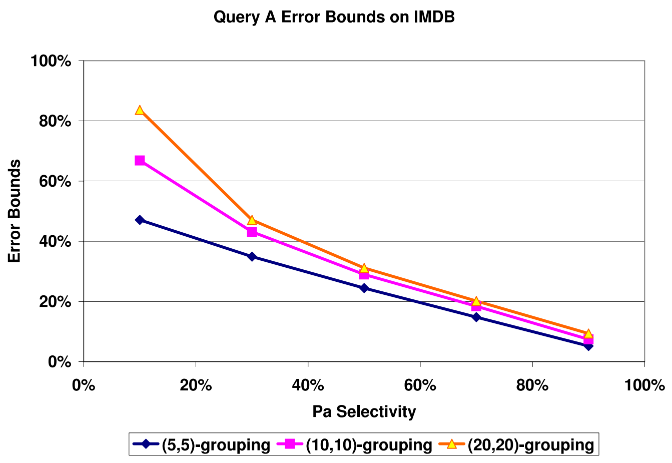
The following parameters can impact query accuracy:

- Group size: We compare approaches from $(k,1)$-, $(1,\ell)$- and (k,ℓ)-groupings. We expect smaller group sizes to offer better accuracy for query answering.
- Selectivity of predicates: More highly selective queries are more likely to touch just a few nodes within a single group, and so lead to wider worst case bounds.
- Grouping formation: We will study the impact of building the groupings based on an arbitrary initial ordering of the nodes, and based on sorting of degree and neighborhood degree, as discussed in Section 4.4. We expect sorted groupings to give better answers when queries have structural predicates based on degree.
- Interaction of grouping and predicates: There may be some implicit correlation between the groups and the query predicates which will improve query answers: e.g. selecting mathematical papers may skew towards fewer authors, while selecting physical sciences papers may skew towards more authors. Groupings based on sorting by structural properties may give better results here.

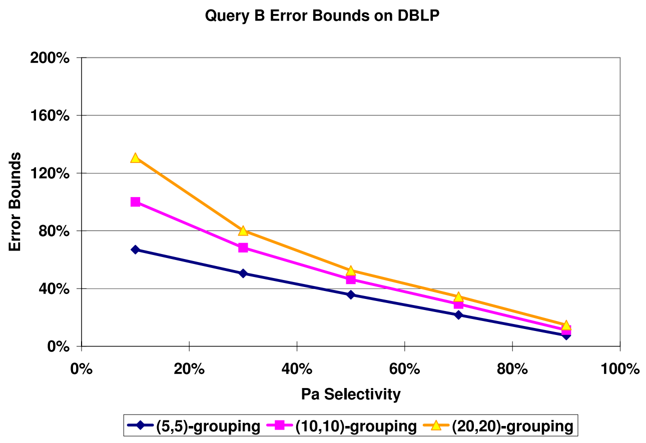
In the following, we show a set of experiments and evaluate the impact on the query accuracy of all the above factors. For our experiments, we found a variety of groupings based on different initial orderings of the data, as described below.



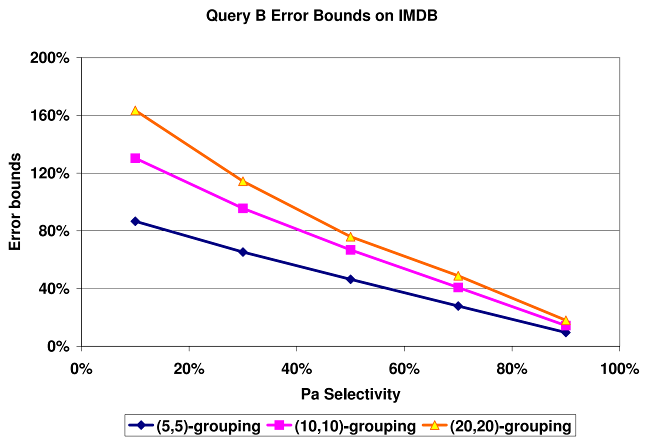
(a) Query A on DBLP



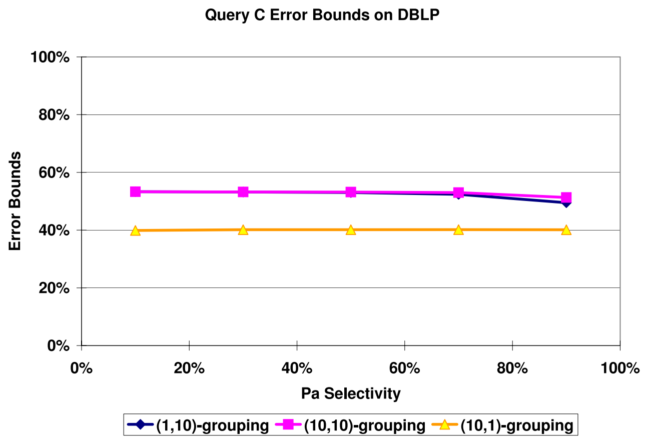
(b) Query A on IMDB



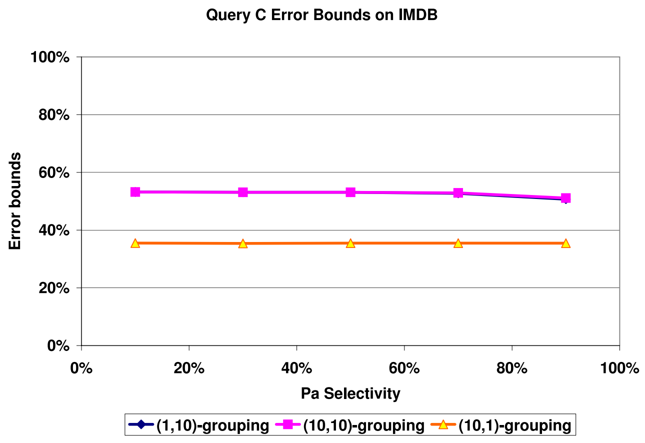
(c) Query B on DBLP



(d) Query B on IMDB

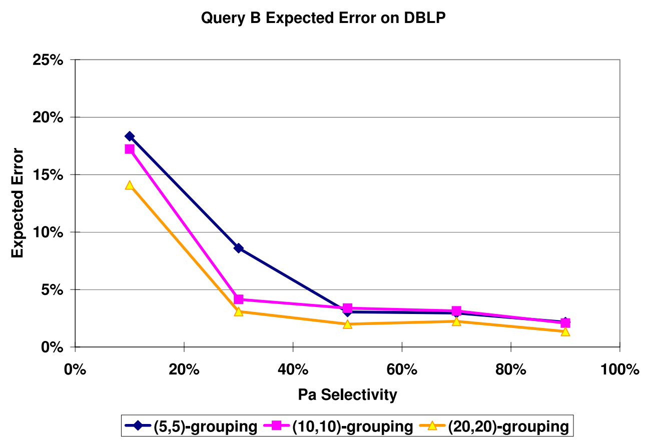


(e) Query C on DBLP, P'_a selectivity 0.8

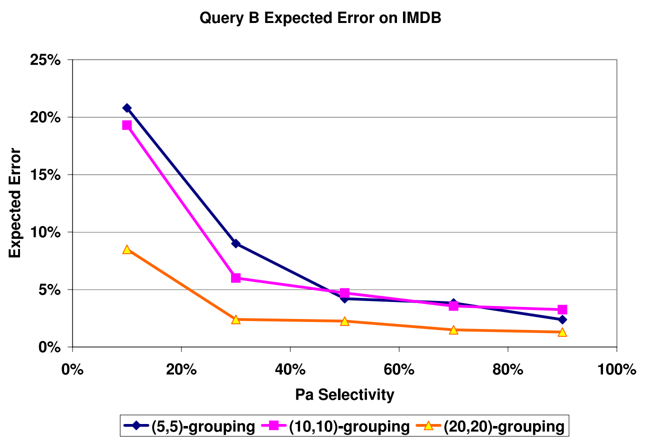


(f) Query C on IMDB, P'_a selectivity 0.8

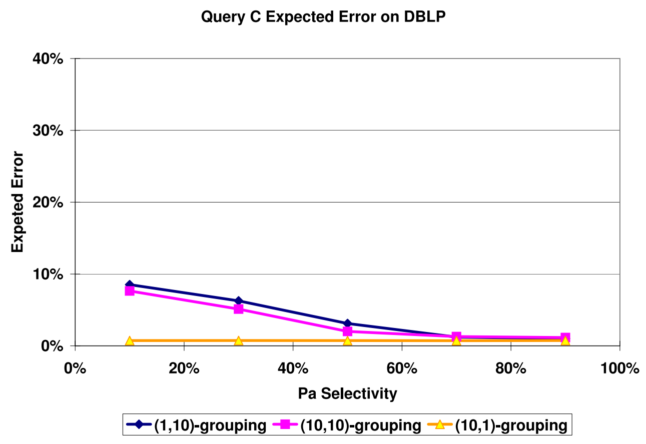
Fig. 12 Impact of query selectivity and group size on three queries



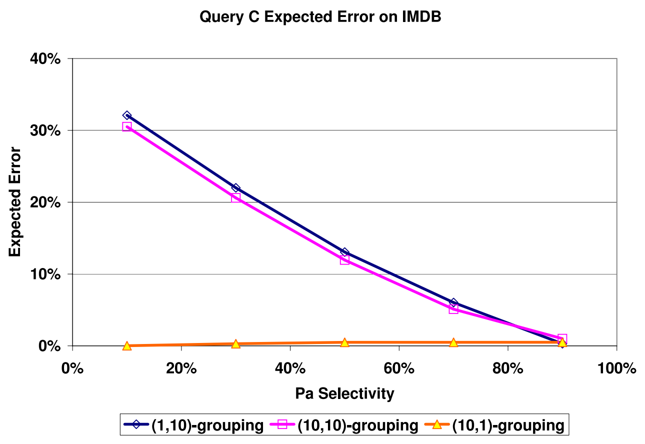
(a) Expected Error on Query B, DBLP data



(b) Expected Error on Query B, IMDB data



(c) Expected Error Query C, P_a' selectivity 0.1, DBLP data



(d) Expected Error Query C, P_a' selectivity 0.1, IMDB data

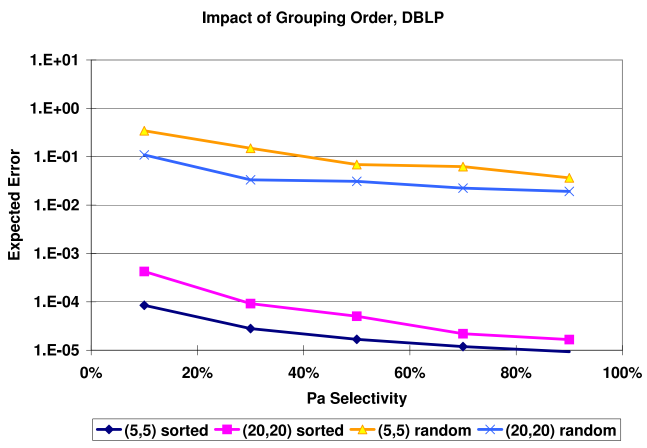
Fig. 13 Query selectivity and group size, expected error

4.7.1 Worst Case Error Bounds.

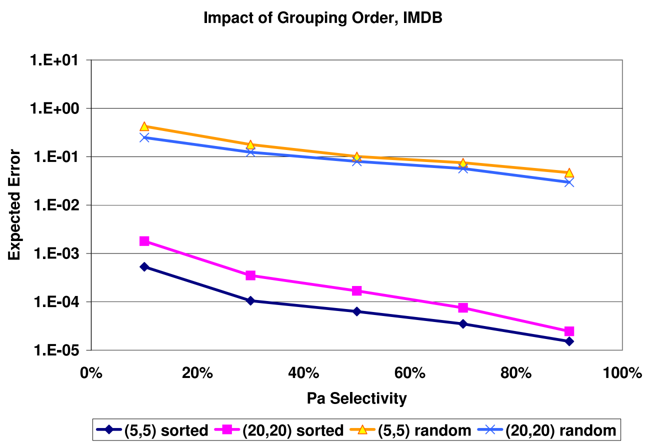
In our first set of experiments, we formed groups of nodes with an initial ordering based on grouping together nodes with the same degree and second-order degree. However, they were not *sorted* by degree, only *grouped* by degree. Figure 12 shows the worst case error bounds for query answering with (k, k)-groupings over the queries A, B, and C. As expected, smaller groupings achieve smaller uncertainty. There is also a clear trend for Queries A and B (Figures 12(a) to 12(d)) that as the selectivity of P_a increases, the accuracy improves. When only a single node in a group is touched by a query, as happens when selectivity is low, it could be any node, and so we have high uncertainty for the aggregate value in the group. But when many nodes are selected in a group, there is less relative uncertainty for an aggregate like sum or average. Further, smaller groups improve the accuracy of the answer: a (5,5) grouping is always better than a (10,10) grouping, which is better than a (20,20) grouping. However, the relation between accuracy and group size is not linear: doubling the group size increases the error by much less than twice. The trend across data sets is similar. Error appears lower in general on the DBLP data, possibly due to lower variation in the number of authors per paper than actresses per movie for query A.

For Query C (Figures 12(e) and 12(f)), there is little variation as P_a's selectivity varies (in this plot, selectivity of P_a' is set to 0.8; similar experiments for other values of P_a' showed the same results). Note that when we have a paper group of size 1, as in the $(10, 1)$-grouping, we can directly select out exactly those papers that meet the predicate, and so have better accuracy compared to other groupings. There is little difference between the $(10, 10)$-grouping and the $(1, 10)$-grouping. This is because P_a' selects most authors, so there is not much benefit from the $(1, 10)$-grouping's ability to eliminate some candidates. When P_a' selects fewer authors, there is a clearer advantage of $(1, 10)$ over $(10, 10)$ grouping. Behavior is fairly consistent over the data sets: for this query, accuracy is slightly better on the IMDB data, but the difference is marginal.

We also studied the time cost of query answering, and the results are shown in Table 2. For all these experiments

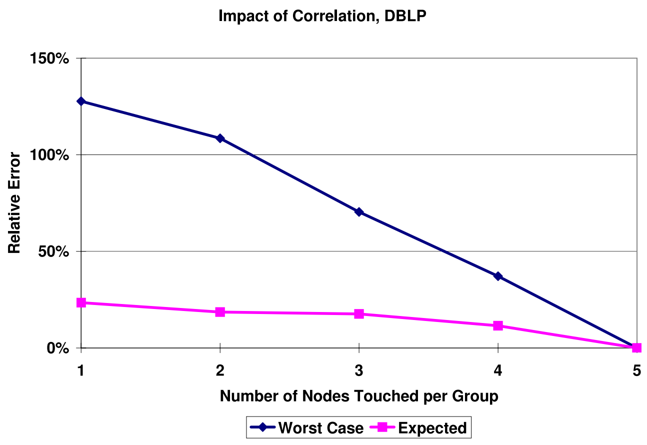


(a) Impact of Grouping Order on DBLP

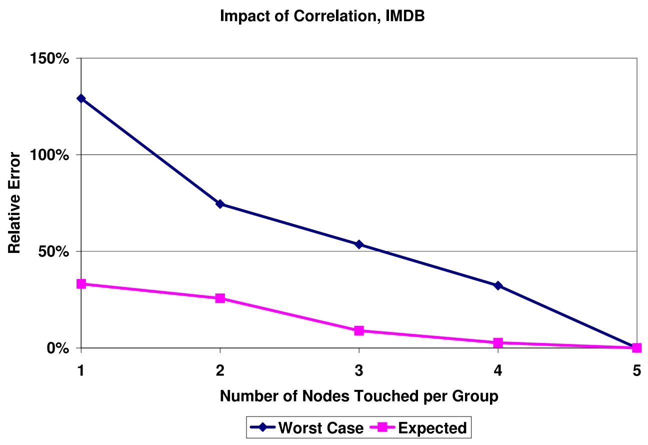


(b) Impact of Grouping Order on IMDB

Fig. 14 Impact of Grouping Order



(a) Correlation between predicate and groups, DBLP



(b) Correlation between predicate and groups, IMDB

Fig. 15 Correlation between predicate and groups

Table 2 Query running time (s)

Dataset	Query A	Query B	Query C	
			$P'_a = 0.8$	$P'_a = 0.1$
DBLP	38.8	13.7	144.6	36.0
IMDB	24.0	7.5	157.7	95.1

we use a (10, 10) grouping and $P_a = 0.5$; other settings were broadly similar. They show that the more complex queries take more time to answer, and show some dependence on the selectivity of the predicates (since groups containing no nodes selected by the predicate can be ignored).

4.7.2 Expected Case Error Bounds.

Although the above worst case bounds show that there can be a wide range between the upper and lower bounds on a query, we show next that the expected bound can give a quite accurate answer. Indeed, on query A, the observed error in using the expected answer was so close to zero that we omit plots since there is little to observe there. Figure 13 shows the expected error on queries B and C. The general trend is again that higher values of selectivity give better accuracy. However, observe that the expected errors are much smaller than the worst case bounds, and do not vary much based on group size—in several cases a larger grouping achieves better *expected* error than a smaller one. This suggests that the tradeoff between privacy and utility can be more complex in practice. On query C, as in the worst case, the expected error is much smaller on $(10, 1)$ than $(1, 10)$ or $(10, 10)$, which are about the same for this (more selective) P'_a; similar results occur for other values of P'_a. Between the two data sets, behavior on query B is quite similar, while for query C, there is an appreciable variation: the error is higher on the IMDB data for the $(1, 10)$ and $(10, 10)$ groupings.

4.7.3 Impact of ordering on grouping.

Query B involves a structural predicate (single author papers), so we compare different choices of grouping in Figure 14. The "sorted" case first sorts the data by degree and second-order degree before finding a safe grouping, while the "random" case picks an arbitrary ordering before finding a safe grouping. We see that there is a very dramatic benefit to having a grouping based on the sorted ordering: two orders of magnitude improvement in the accuracy on data sets. This is because most groups now contain papers with the same number of authors, meaning the contribution to the aggregate can be found exactly for those groups, and only the few that remain contribute to the uncertainty. The same behavior occurs for both data sets, with the same factor of improvement observed, although accuracy is consistently better for DBLP than for IMDB. Lastly, note that the best case occurs for the smaller groups ((5,5) compared to (20,20)) when the input to the grouping procedure is sorted by degrees.

We further investigate the impact of the correlation of the predicate with the grouping. For query B, we construct an artificial predicate P_a which selects the same number of total papers, but touches a variable number of papers in each paper group within a $(5,5)$ grouping. Figure 15 shows that as the number of papers touched in each group increases, both the expected and worst case bounds improve up to the point when all papers are selected in a group, the aggregate query is answered with perfect accuracy. This shows that if we can anticipate the kinds of structural predicates that end users will want, then we can improve the utility of the published data without compromising the privacy.

4.7.4 Grouping on Dense Data

As discussed in Section 4.2, a safe grouping is easy to find for sparse datasets. The previous experiments were on relatively sparse datasets. In this section we examine the performance of our algorithms when the given dataset is more dense. We study the FIMI dataset described in Section 2.4.

We first analyze the difficulty of forming a safe grouping on the transactions. We ran the grouping algorithm, seeking groups of size 8, 10 and 12. The running time was 828s, 936s, and 977s respectively. This demonstrates that that even for such a dense graph, our algorithm can still find a safe grouping efficiently. However, it is not possible to find much larger groups, since there is one item which appears in 823 transactions. Since the safety condition allows each item to be linked to at most one transaction in any group of transactions, each of these 823 transactions must be placed in separate groups, which is not possible for group sizes greater than 12. This is a limitation of demanding this level of safety: alternate definitions of safety might allow larger groupings, but would have different privacy implications.

As a second measure of the difficulty of finding safe groupings, we also counted the number of transactions which could not be placed in their "first choice" group: that is, the number of cases where the transaction cannot be placed in the first group which is tried, and a later group has to be used. For groupings of size 8, 10, and 12, out of the 10000 transactions, the numbers of nodes that failed the first trial are 7524, 8420 and 9051 respectively. This demonstrates that as group size grows larger, it becomes harder to find a group which satisfies the safety condition.

We now discuss finding groupings of the items. Across all transactions, one item appears in common with 768 others. Moreover, there are many other such items with high degree and highly dense interaction pattern. So it is not possible to find a grouping into groups of size 2 which meets the safety condition. However, it can be argued that in this example, it is more important to find groupings over the transactions. Note that the implication for this is that however we group the items, there is likely to be some transaction where the grouping reveals that all items in the item group are linked to that transaction. This is an inherent problem with grouping this data, rather than with our approach to grouping.

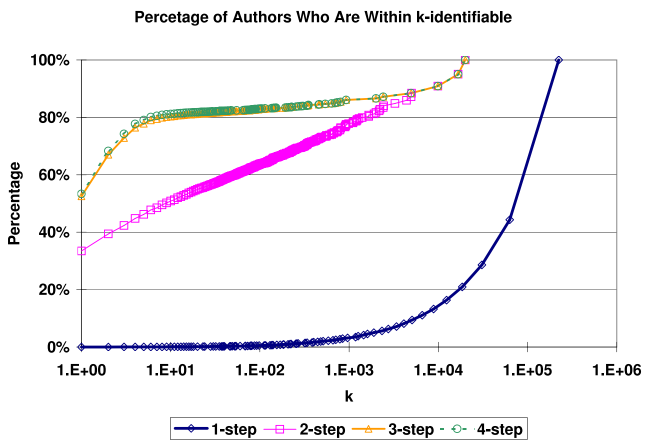
5 Unions of groupings

In this section, we consider the impact of publishing multiple groupings of the same graph. This allows a broader class of queries to be answered with perfect accuracy, but is open to stronger classes of attack based on the graph structure. Recall that publishing the fully-censored (m,n)-grouped graph allows type-0 queries to be answered exactly, but gives us no handle to answer other query types with certainty. As observed in Section 4.6, publishing $(1,\ell)$ or $(k,1)$ grouped graphs offers greater utility for a variety of queries while preserving the privacy of associations.

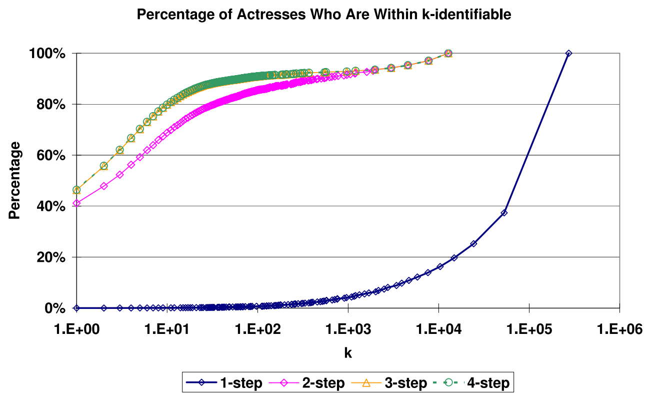
We can give greater utility if we fully censor only one side of the bigraph, leading to $(1,n)$ and $(m,1)$-groupings. In this case, V (respectively W) is preserved perfectly, while all of W (resp. V) is placed into a single group. This simplifies the information we have to publish:

Definition 6 Given a bipartite graph $G = (V,W,E)$, its $(m,1)$-grouping is defined by $G_V(V,W,H_V(E))$ where: H is an arbitrary injective function mapping from $V \cup W$ onto the integers, and $H_V(E) = \{(H(v_i),w_j)|(v_i,w_j) \in E\}$. Similarly, its $(1,n)$-grouping is defined by $G_W(V,W,H_W(E))$, where $H_W(E) = \{(v_i,H(w_j))|(v_i,w_j) \in E\}$.

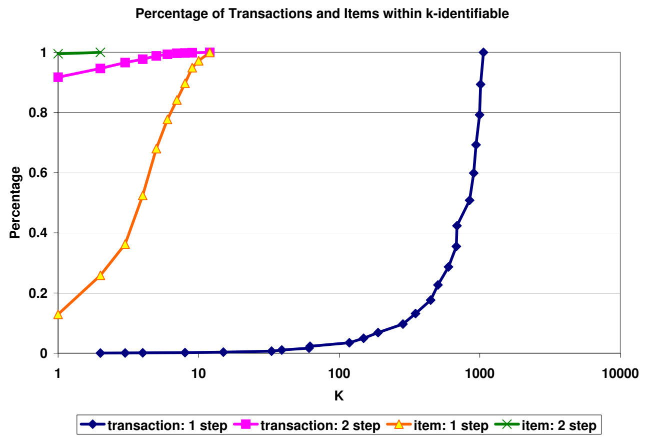
Observe that if we apply a $(1,n)$ grouping to a graph that is already $(m,1)$ grouped, then the result is the (fully



(a) Identifiability of authors



(b) Identifiability of actresses



(c) Identifiability of transactions and items

Fig. 16 Attacking a $(m, 1) \cup (1, n)$-grouping

censored) (m, n) grouped graph. Given a $(m, 1)$-grouping of a graph, we can answer certain query types with perfect accuracy. In particular, we can answer any query of type 1, provided that the predicate P_a is over nodes of W. Similarly, we can answer any query of type 1 where the predicate P_a is over nodes of V given the $(1, n)$-grouped version. Privacy of either graph is also guaranteed: without any additional knowledge, one cannot infer the identity of any of the censored nodes. However, we cannot give any high-quality answer to such queries if we only have the $(m, 1)$-grouped graph.

So it seems that publishing both, as a $(1, \ell) \cup (k, 1)$ grouping of a graph, is desirable, since any query of type 1 could be answered exactly. But while either version of the graph in isolation is resilient against attack, publishing both allows an attacker to combine the information in the static attack model (Definition 1), as they now have information not previously available to them. If a customer has bought more products than any other, she can be identified *statically* from the $(1, \ell)$ graph, i.e. with no background knowledge about degrees or other data. From its unique degree, the same node can be located in the $(k, 1)$-grouped graph, revealing all the products bought by that customer. This attack applies even over $(m, 1) \cup (1, n)$-grouping, which gives the most privacy in this class.

More strongly, if certain types of mappings between the isomorphic $(m, 1) \cup (1, n)$-grouped graphs can be found, the original data can be recovered. This problem is related to, but distinct from, the (well-studied) graph isomorphism problem. Some information will remain private: for example, when there are customers who have only ever bought a single product unique to them, their associations cannot be recovered from $(k, 1) \cup (1, \ell)$-grouped graphs. In that example, it is easy to find a valid isomorphism over these nodes, but there are many other valid isomorphisms that associate different customers with different products, so the attacker cannot be sure which was the original mapping. But if the attacker can find node or edge pairs that must be *uniquely* mapped to each other in *every* isomorphism (i.e. every possible world that is consistent with the published information), their privacy is compromised. Clearly, the amount of privacy that remains is input dependent: data consisting solely of nodes with degree 1 is secure; but if each node has a unique degree then total re-identification is trivial. On realistic data, the truth lies somewhere in between.

5.1 Experimental Analysis of Privacy.

We attack $(m,1) \cup (1,n)$-grouped graphs, based on finding matching pairs of nodes between the two graphs. Each node in the fully censored graph is given a compact *signature*. Initially, the signature of every node is a default value, say 0, since there is no *a priori* way of telling them apart. We choose the 1-step signature of a node to be the degree of that node. Given a node, its *next-step signature* is formed by concatenating its current signature with the signatures of all its neighbors in the graph, and sorting this set lexicographically. Once next-step signatures are found for all nodes in the graph, they can be compactly relabeled (since there can be at most n different signatures for n nodes). By this construction: (a) If two nodes have different signatures then they cannot be matched in any isomorphism—since the signature canonically encodes features of the neighborhood of a node, different signatures entail non-isomorphic neighborhoods. (b) Since the process is entirely deterministic, each node will obtain the same signature every time the procedure is run on the graph. As a result, if a node receives a signature that is not shared by any other node, then this node must be uniquely matched in any isomorphism. Moreover, it can be matched to the unique node with the same signature in an isomorphic copy. Note that the implication is only one way: the guarantee is that if signatures are unique then nodes can be uniquely matched, and not vice-versa. Schemes which build signatures for edges instead of for nodes are also feasible; the details are quite similar, and we present results only for the node-based scheme for brevity.

We therefore study the effectiveness of this attack on the anonymized data. We apply this signature scheme on the $(m,1) \cup (1,n)$-grouped graphs, and measure how many nodes are uniquely identified, and how many fall into equivalence classes of size 2, 3, 4 etc. The cumulative distribution over such classes of authors in the DBLP dataset, actresses in the IMDB data, and transactions and items in the FIMI data are shown in Figure 16. Multiple steps of signature computation were performed, but for all datasets, no improvement was seen after the fourth iteration, and there is only limited difference from the third to fourth step. A 4-step signature is sufficient to identify half the authors and almost half the actresses uniquely. Only about 20% of authors and actresses are in equivalence classes of 10 or larger. Such privacy levels are weak for many typical applications. For the FIMI transaction data the result is even starker: although no transactions are uniquely identified by a one step signature, a 2-step signature is sufficient to uniquely identify 91.8% of the nodes. With this signature, the largest equivalence class has only twelve members. For the items, a two step signature allows 862 out of the total 866 nodes can be uniquely identified. So we conclude that $(1,\ell) \cup (k,1)$-groupings should be avoided. The single groupings discussed in the previous section offer much stronger privacy guarantees while allowing queries to be answered accurately. This highlights that much care is needed for problems of anonymization, and that enhancements designed to improve utility run the risk of opening themselves to very effective attacks on privacy.

6 Related Work

The problem of how to anonymize and publish data for others to analyze and study has attracted much study in recent years. Starting with the pioneering work of Sweeney and Samarati on k-anonymization [16,15], the core problem of anonymizing data tables has led to new techniques and definitions such as ℓ-diversity [11], (α,k)-anonymity [18], t-closeness [10], (c,k)-safety [12], and anonymization via permutation [23,20]. Our attempts to apply some of these methods to our problem in Section 3 either failed to give the required privacy or yielded results with very low utility.

There has been considerable recent interest in anonymizing data which can be represented as a graph, motivated by wanting to publish social network data. Backstrom *et al.* [1] consider attacks on publishing such data with identifiers removed (the "fully censored" case). They study both active attacks, in which the attacker is allowed to insert a number of nodes and edges into the graph before it is published, and passive, where the attacker learns all the edges incident on a set of linked nodes. In both cases, a large enough known subgraph can be located in the overall graph with high probability, and hence information can be learnt about connections between nodes. However, as here, *nothing* is learnt about connections between nodes that are *not* incident on edges known to the attacker.

Hay *et al.* [8] analyze what privacy is present inherently within the structure of typical social networks, by measuring how many nodes have similar or identical neighborhoods (based, e.g. on degrees of nearby nodes). This is similar to the attack we studied in Section 5.1. They analyze what additional privacy is gained by deleting and then randomly inserting up to 10% of edges, but observe that such modification can significantly alter graph properties. Similarly, Zhou and Pei [25] define privacy so that each node must have k others with the same (one-step) neighborhood characteristics, and measure the cost as the number of edges added, and number of node label generalizations. Korolova *et al.* [9] analyze attacks in a different model, where the attacker can "buy" information about the neighborhood of certain nodes, and wishes to minimize their cost to learn the graph. Zheleva and Getoor [24] study the effectiveness of machine learning techniques to infer sensitive links which have been erased, given a graph in which non-sensitive links have been anonymized. They consider anonymizations based on grouping nodes: randomly deleting some non-sensitive edges; reporting only the number of edges between groups (similar to Section 3.3); and just reporting whether two groups have any edges. They do not consider our approach of retaining the graph structure but hiding the mapping from entities to nodes. Our work differs from prior work essentially because we focus on a different region of the privacy-utility tradeoff: we consider settings where releasing the unlabeled graph is permitted, but lacks utility, whereas prior work does not allow such release.

More recently, Hay *et al.* [7] extended their study of reidentification of graph data, and proposed forming nodes into groups and revealing only the number of edges between pairs of groups. Given the same sets of groups of nodes, their approach would entail a much larger number of possible worlds that could correspond to the published data. This approach is similar to the initial permutation approach considered and rejected due to its inability to retain fundamental graph properties; the work of Hay *et al.* differs in the way that groups are formed and edges are permuted, and so seems able to attain better, but still degraded, fidelity for (simple) graphs. Extending this approach, Campan and Truta propose building "clusters" (groups) of nodes, and revealing only the number of edges within a group and between pairs of groups [4]. The nodes have additional properties, which are generalized so that all nodes in the same cluster have the same generalized representation.

Also relevant is work which considers relations with many sensitive attributes, since such data is often effectively represented in graph form. Nergiz *et al.* [14] mention the shortcomings of representing and anonymizing bitmap representations of relational data, which we argue is also insufficient for graph data in Section 3.2. Closest to our work in setting is recent work by Ghinita *et al.* [6] on anonymizing sparse high-dimensional data (since a bipartite graph can be seen as defining such a sparse relation). Their approach is to extend known permutation based methods [23,20] to improve utility. In their data, sensitive attributes are rare, so they can ensure at most one sensitive attribute in each group of k individuals; in contrast, in our setting, every attribute (association) is sensitive and so we cannot apply their method. Moreover, [6] does not consider graph properties of the data, which we take care to preserve. Work on ℓ-diversity briefly considers the issue of multiple sensitive attributes, and concludes that much larger groups would be needed to guarantee privacy [11]. The crucial difference that allows our techniques to succeed is that although we have a large number of sensitive attributes (e.g. all customers) in graph data, the graph is sparse, so these can be hidden amongst many possible associations.

Lastly, our work can be compared to that designed to anonymize transactional or set data, such as recent contributions by Terrovitis *et al.* [17], and Xu *et al.* [22]. There, the problem is to apply a suitable anonymization to a dataset of transactions, where each transaction is a set of items connected to an individual. Clearly, this scenario can also be modeled as a bipartite graph, as shown with our continued example of customers and products. However, the power of the adversary, and hence the goal of the anonymization is different in [17,22]. There, it is assumed that an attacker is able to observe some number of items belonging to an individual, and wishes to infer other (private) items in their transaction. Our methods were not designed with this model of attack, and as a result are not comparable (likewise, the methods of [17,22] were not designed for bipartite graph data, and as such do not adequately preserve graph structure information). Nevertheless, due to the similar motivations, it will be of interest to find a common framework for these results, and to extend the grouping approach we develop here to give guarantees against this model of adversary.

7 Concluding Remarks

We have considered the problem of anonymizing data in the form of bipartite graphs, and shown that methods based on finding safe (k, ℓ)-groupings are effective at securing published data against a variety of attacks. We have shown how to answer queries for various natural classes of aggregates, but it remains to automatically rewrite arbitrary queries to give upper, lower and expected bounds on safely grouped graphs. It is also of interest to study advanced query types, such as join-style queries over longer edge paths.

We have assumed that full information can be revealed about entities, but the mapping from entities to nodes in a graph must be masked. Other models may be needed if we wish to anonymize both entities *and* the associations between them. Our focus has been on data that can be represented as a bipartite graph linking two types of entity. It is natural to also study arbitrary graphs over a single type of entity, i.e. social network graphs [1,8,9]. There have been large increases in the quantity of data representing interactions in social networks being collected in recent years. This has led to greater interest in applying anonymization techniques to such data to allow sharing and analysis without compromising the privacy of the individuals whose data is stored by the networks. In our ongoing work, we plan to study the extent to which our methods can be generalized and extended to accurately model and anonymize this kind of data. Some initial results in this direction are presented in [3].

References

1. L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore are thou R3579X? Anonymized social networks, hidden patterns and structural steganography. In *International Conference on World Wide Web (WWW)*, 2007.
2. J. Bennett and S. Lanning. The Netflix prize. In *KDDCup Workshop*, 2007.
3. S. Bhagat, G. Cormode, B. Krishnamurthy, and D. Srivastava. Class-based graph anonymization for social network data. Technical report, AT&T Labs–Research, 2008.
4. A. Campan and T. M. Truta. A clustering approach for data and structural anonymity in social networks. In *International Workshop on Privacy, Security and Trust in KDD (PinKDD)*, 2008.
5. M. R. Garey and D. S. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.
6. G. Ghinita, Y. Tao, and P. Kalnis. On the anonymization of sparse high-dimensional data. In *IEEE International Conference on Data Engineering*, 2008.
7. M. Hay, D. Jensen, G. Miklau, D. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. In *International Conference on Very Large Data Bases*, 2008.
8. M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava. Anonymizing social networks. Technical Report 07-19, University of Massachusetts Amherst, 2007.
9. A. Korolova, R. Motwani, S. Nabar, and Y. Xu. Link privacy in social networks. In *ACM Conference on Information and Knowledge Management (CIKM)*, 2008.
10. N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *IEEE International Conference on Data Engineering*, 2007.
11. A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. In *IEEE International Conference on Data Engineering*, 2006.
12. D. J. Martin, D. Kifer, A. Machanavajjhala, and J. Gehrke. Worse-case background knowledge for privacy-preserving data publishing. In *IEEE International Conference on Data Engineering*, 2007.
13. A. Narayanan and V. Shmatikov. How to break anonymity of the Netflix prize dataset. Technical Report arXiv:cs/0610105v1, arXiv, 2006.
14. M. E. Nergiz, C. Clifton, and A. E. Nergiz. Multirelational k-anonymity. In *IEEE International Conference on Data Engineering*, 2007.
15. P. Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.
16. L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based systems*, 10(5):557–570, 2002.
17. M. Terrovitis, N. Mamoulis, and P. Kalnis. Privacy-preserving anonymization of set-valued data. In *International Conference on Very Large Data Bases*, 2008.
18. R. Wong, J. Li, A. Fu, and K. Wang. (α, k)-anonymity: An enhanced k-anonymity model for privacy-preserving data publishing. In *ACM SIGKDD*, 2006.
19. R. C.-W. Wong, A. W.-C. Fu, K. Wang, and J. Pei. Minimality attack in privacy preserving data publishing. In *International Conference on Very Large Data Bases*, 2007.
20. X. Xiao and Y. Tao. Anatomy: Simple and effective privacy preservation. In *International Conference on Very Large Data Bases*, 2006.
21. X. Xiao and Y. Tao. M-invariance: towards privacy preserving republication of dynamic datasets. In *ACM SIGMOD International Conference on Management of Data*, 2007.
22. Y. Xu, K. Wang, A. W.-C. Fu, and P. S. Yu. Anonymizing transaction databases for publication. In *ACM SIGKDD*, 2008.
23. Q. Zhang, N. Koudas, D. Srivastava, and T. Yu. Aggregate query answering on anonymized tables. In *IEEE International Conference on Data Engineering*, 2007.
24. E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationships in graph data. In *International Workshop on Privacy, Security and Trust in KDD (PinKDD)*, 2007.
25. B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *IEEE International Conference on Data Engineering*, 2008.