On Disclosure Risk Analysis of Anonymized Itemsets in the Presence of Prior Knowledge [1]

LAKS V.S. LAKSHMANAN
University of British Columbia
RAYMOND T. NG
University of British Columbia
and
GANESH RAMESH
Microsoft Corporation

Decision makers of companies often face the dilemma of whether to release data for knowledge discovery, vis a vis the risk of disclosing proprietary or sensitive information. Among the various methods employed for "sanitizing" the data prior to disclosure, we focus in this paper on anonymization, given its widespread use in practice. We do due diligence to the question "just how safe is the anonymized data". We consider both the scenarios when the hacker has no information, and more realistically, when the hacker may have partial information about items in the domain. We conduct our analyses in the context of frequent set mining and address the safety question at two different levels: (i) how likely are the identities of individual items cracked (i.e. reidentified by the hacker), and (ii) how likely are sets of items cracked. For capturing the prior knowledge of the hacker, we propose a *belief function*, which amounts to an educated guess of the frequency of each item. For various classes of belief functions, which correspond to different degrees of prior knowledge, we derive formulas for computing the expected number of cracks of single items and for itemsets, the probability of cracking the itemsets. While obtaining the exact values for the more general situations is computationally hard, we propose a series of heuristics called the *O-estimates*. They are easy to compute, and are shown to be fairly accurate, justified by empirical results on real benchmark datasets. Based on the O-estimates, we propose a recipe for the decision makers to resolve their dilemma. Our recipe operates at two different levels, depending on whether the data owner wants to reason in terms of single items or sets of items (or both). Finally, we present techniques using which a hacker's knowledge of correlation in terms of co-occurrence of items can be incorporated into our framework of disclosure risk analysis and present experimental results demonstrating how this knowledge affects the heuristic estimates we have developed.

Categories and Subject Descriptors: E.m [**Data**]: Miscellaneous

General Terms: Security, Algorithms

Additional Key Words and Phrases: Disclosure Risk, Anonymization, Prior Knowledge, Frequent Itemsets, Hacker, Sampling, Bipartite Graphs, Belief Function, Matching, Correlation

Author's address: Laks V.S. Lakshmanan, Raymond T. Ng, Department of Computer Science, University of British Columbia, 201-2366 Main Mall, Vancouver B.C. V6T 1Z4, Tel.: +1-604-8223153, Fax: +1-604-8225485, E-mail: {laks,rng}@cs.ubc.ca, Ganesh Ramesh, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, USA, E-mail: ganrame@microsoft.com

1. INTRODUCTION

Privacy-preserving data mining has attracted a lot of attention in recent years. The primary objective is to strike a balance between two opposing forces: the urge to mine data to gain further insights and knowledge, versus the responsibility to protect the privacy and identity of individuals (e.g., patients, travelers). These two forces manifest themselves in similar ways in business situations. Decision makers of companies often face the dilemma of whether to release data for knowledge discovery, given the risk inherent in disclosing proprietary/sensitive information (e.g., identities of the better-selling products) to the public, particularly to potential competitors. Consider the following scenarios.

Mining as a service: A company, with insufficient expertise in data mining, wants to hire a data mining service provider to mine its data[2]. While there is legal protection (e.g., non-disclosure agreements), the company still legitimately worries about its data being leaked out somehow, and such leakages are often hard to detect.

Mining for the common good: A company may want to participate in a consortium, which involves sharing of data. The motivation of pooling data together is to gain in scale and in diversity (e.g., geographical variations, demographic differences). The dilemma is that partners of the consortium may one day become competitors, or work with competitors.

One common approach to handling these situations is to release *transformed* data. Among the well-known transformation techniques, *anonymization* is arguably the most common. That is, if the objects in the domain are originally identified by their social security number, product number, etc., these objects are now identified by a generated number, typically as simple as a positive integer. Compared with other transformation techniques, anonymization is simple to carry out, as mapping objects back and forth is easy. Another advantage of anonymization is that it does not perturb data characteristics. For both the above scenarios, changing the data characteristics may affect the outcome too much that it defeats the original purpose of releasing the data. Note that there are alternatives and more sophisticated techniques (e.g. k-anonymization [L. Sweeney 2002]); and we are *not* recommending the use of anonymization instead of those more sophisticated approaches. Our study here is simply based on the observation that anonymization is already widely used in practice. One prime example is clinical trial studies for new drugs in the medical and pharmaceutical domain. Even though the US Food and Drug Administration guidelines are well-known to be strict, anonymization (or de-identification) is still considered adequate in the clinical trial circles for protecting the privacy of the patients participating in the studies. [3]

[2]An alternative is for the company to purchase off-the-shelf data mining software. However, data mining is a process that involves not only the software or the algorithms, but also the expertise in understanding and executing the many steps required – namely, from cleaning the data, preprocessing, to feature selection and tweaking the parameters so that the results are interpretable. Thus, many companies would consider the out-sourcing option to be more timely, reliable and hence cost effective.

[3]Refer to Part 21: Protection of Privacy in Title 21, Chapter 1, Food and Drugs Administration at http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?CFRPart=21.

The immediate question, however, is: *Just how safe is the anonymized data?* where "safe" is interpreted as protection of the identities of the objects – either as individual items or item sets. Hereafter, we use the term data "owner" to refer to the party that owns the original data, and the term data "hacker" to refer to the party that tries to identify or illegally reveal the true identities of anonymized objects. We use the term "cracks" to refer to the objects whose identities are detected or inferred by the hacker. The reason why we analyze cracking of items is that this would enable the hacker who has access to the anonymized database to identify sensitive patterns in the original data. We are interested in cracks at the level of single items as well as at the level of item sets, since a set S as a whole may be cracked even when no item in the set is individually cracked by a hacker. An answer to the question of safety is complicated by an issue often overlooked in the literature: *How much partial information does the hacker have?* The assumption that there is no partial information out there is simply unrealistic in this Internet era. Furthermore, as illustrated in the two application scenarios above, a hacker may be from a competitor or a rival company. The hacker may use his/her knowledge from his/her own data or data from similar sources, to infer and gain knowledge about the anonymized data. It has been recognized that incorporating partial knowledge in analyzing security is an important open research problem. For instance, Yang and Li consider prior knowledge captured in functional dependencies in secure XML publishing [X. Yang and Chen Li 2004]. Thus, the safety question becomes: *Just how safe is the anonymized data in the presence of partial information?*

One source of partial information is when the hacker has access to similar data. In this paper, we develop a framework to answer the safety question by investigating various abstractions of similarity. Using frequent set mining as an illustrative example, we model the hacker's partial information in the form of a *belief function* which represents the hacker's educated guess about the frequency of each item from such similar data. For each of these abstractions, we explore how to determine the percentage of cracks when the hacker has access to the anonymized database and partial information. In a preliminary version of this paper [Laks V.S. Lakshmanan et al. 2005], we proposed the first framework for analyzing the safety of anonymized data and developed methods for estimating the expected number of cracks for individual items in the presence of partial background knowledge by the user. In this paper, while we give a brief summary of those results, our main focus in on *itemsets*, which are significantly more complex than individual items. Specifically, we address the question: given an itemset, how likely is it to be cracked by the hacker given his/her partial knowledge about the item domain. It turns out the model and recipe presented in [Laks V.S. Lakshmanan et al. 2005] are inadequate for handling itemsets. Accordingly, we extend the model and present a new recipe for use by the data owner to help them decide whether they should feel comfortable releasing the data. Finally, we also show how our existing framework of disclosure risk analysis can handle a hacker's prior knowledge of co-occurrence of items. Our contributions are the following:

—We first analyze two extremes of similarity (Section 3): When the hacker has access to no data and when the hacker has access to "almost" identical data.

This is captured in the form of *ignorant* and *compliant point-valued* belief functions respectively. For both cases, we present exact formulas for determining the expected number of cracks.

—While the extreme cases may be highly unlikely to occur in practice, their analysis, nevertheless, is useful in building the necessary tools for more realistic scenarios. We model the third abstraction of similarity where the partial information is captured in a *compliant interval(-valued)* belief function (Section 4), which specifies a range of frequency values for each item. Each range is assumed to contain the true frequency (hence compliant). This abstraction models the case when the hacker has information about the true frequency of every item but is not sure about its exact value (hence a range). The fourth abstraction of similarity captures the most general notion of the hacker's partial information: α-compliant belief functions where the belief function only guesses the right ranges for a fraction α of items ($0 \leq \alpha \leq 1$).

—Not surprisingly, exactly determining the expected number of cracks for interval belief functions turns out to be a hard problem, and even known estimation algorithms have too high a complexity to be practical. Companies which need to out-source data mining are most likely to be interested in only light-weight tools for assessing the risk of releasing the data. Thus, it is our design objective to develop light-weight tools for this purpose. To this end, we develop the O estimate for the single items (Section 4.1) and the OS estimate for general itemsets (Section 4.3). We show by evaluating through a systematic set of experiments on benchmark data (Section 6) that these estimates are practically accurate for the various similarity abstractions.

—The notion of the expected number of cracks may not be easy to be interpreted. Thus, we translate the expected number of cracks to a level of compliancy that we believe is easier for the data owner to make a decision. We propose lightweight recipes, using which a data owner can determine the risk of releasing anonymized data (Section 5). We show the effectiveness of these recipes (Section 6.2.2).

—Finally, we show how our disclosure risk framework can handle a hacker's prior knowledge of correlated pairs of items in the data. We show experimentally how it impacts the heuristic estimates and present an extended recipe to handle this kind of prior knowledge.

1.1 Related Work

Given that we analyze the disclosure risk in releasing anonymized data, the following bodies of work are most relevant. First, anonymization is akin to the use of substitution ciphers in secure communication. The idea there is to map the original alphabet to a so-called substitution alphabet in a one-to-one way. This has been in use since a long time ago. More details on substitution ciphers and their weaknesses can be found in [Alan G. Konheim 1981]. The risk analysis conducted in this paper is applicable to substitution ciphers as well.

Second, there is a substantial body of work on k-anonymization and its variants [P. Samarati and L. Sweeney 1998; L. Sweeney 2002; Gagan Aggarwal et al. 2005; A. Meyerson and R. Williams 2004]. See [Gehrke 2005] for a survey of more recent work in this area. The basic idea is to generalize (and suppress) data records in

such a way that each data item becomes indistinguishable from at least $k-1$ other items in the generalized data set. While this model is similar in spirit to our notion of anonymization, it perturbs the data (in the simplest case, making more than one item indistinguishable from each other), thus making reconstruction of patterns difficult. In any event, this paper does not necessarily advocate anonymization as the best method to limit disclosure. Rather, based on the observation that anonymization is one of the most commonly used approaches, our work gives due diligence to the analysis of the risk of releasing anonymized data. Furthermore, our analysis addresses the often overlooked issue of considering whatever partial information that the hacker may possess.

Third, the problem of *record linkage* has received significant attention. Though similar in spirit to disclosure risk, the problem deals with how records in databases can be linked to individuals based on other information about these individuals. For more detailed results, the reader is referred to the studies by Winkler in [William E. Winkler 1993; 1994; 1999] and the work by Domingo-Ferrer et. al. in [Joseph Domingo-Ferrer and Vincent Torra 2002; Joseph Domingo-Ferrer et al. 2002]. While the problem of record linkage uses statistical methods to re-identify individuals with masked records, the framework of using bipartite graphs to measure the risk of disclosing individual items in the data is not directly evident from this body of work. More recently in [Stephen E. Fienberg and A.B. Slavkovic 2005], the authors study the implications of selectively releasing association rules on the disclosure risk of the original dataset and provide statistical and algebraic geometric connections. While these implications are couched in terms of bounds, they do not consider how these bounds can be converted into procedures that will help the data owner assess the risk in disclosing these rules. Our approach differs from this work in two ways: (1) We consider the risk when the entire dataset is disclosed in anonymized form and not just patterns from this dirtiest. (2) We are interested in how our analysis can translate into simple decision making procedures that can help a data owner assess the disclosure risk.

Besides the above lines of work, there is a body of work of peripheral relevance to this paper. We briefly discuss them below. Several approaches to privacy-preserving data mining are based on perturbing the underlying data for enhancing privacy protection. E.g., privacy has been studied in the context of association rule mining [Vassilios S. Verykios et al. 2004; Alexandre Evfimievski et al. 2004; Murat Kantarcioglu and Christopher Clifton 2004]. In [Alexandre Evfimievski et al. 2004], Evfimievski et. al. propose a framework for mining association rules in which the data items in transactions are randomized to preserve privacy of individual transactions. They analyze the nature of privacy breaches caused by using the association rules discovered from this database and propose randomization operators to limit such breaches. The problem of hiding association rules by transforming the input database is studied by Verykios et al. in [Vassilios S. Verykios et al. 2004]. The authors are interested in modifying the input database such that a given set of associations is hidden in the transformed database. Techniques like removing items from transactions, adding new items to transactions are used. While their problem of transforming data shares some commonalities with our work, they modify the frequency of items in the original data. Furthermore, their studies do not deal with

the hacker's possible prior knowledge.

In [Rakesh Agrawal and Ramakrishnan Srikant 2000], Agrawal and Srikant propose an approach for privacy preserving classification that is based on mining on perturbed data, with the perturbed distribution closely matching the real distribution. Furthermore, Agrawal and Aggarwal in [Daksha Agrawal and Charu C. Aggarwal 2001] discuss an expectation maximization algorithm which provides robust estimates of the original distribution based on perturbation and provides some interesting results on the relative effectiveness of different perturbing distributions in terms of privacy. In [Vijay S. Iyengar 2002], Iyengar uses the approaches of suppression and generalizations to satisfy privacy constraints. The trade-off between privacy and information loss in the specific context of data usage is considered, and the search for the optimal trade-off is considered as an optimization problem for which a genetic algorithm framework is used to search for a solution. In [Charu C. Aggarwal and Philip S. Yu 2004], Aggarwal and Yu use an approach based on *condensation groups* to model indistinguishability of data records and use it to create anonymized data that has similar characteristics to the original multidimensional dataset and apply it to the classification problem.

In sum, all these studies focus on perturbing the data so that the results of mining the perturbed data remain similar to the original data. Our work here is very different in that it gives an analysis of the risk of releasing anonymized data.

The security problem in statistical databases deals with protecting the database from returning information about an individual or answering a sequence of queries from which individual information can be deduced, where the statistical databases allow only queries that retrieve statistical information (like sum, average, median) of certain subsets of records.

The various approaches used by the security control methods are categorized in the survey by Adam and Wortmann [N.R. Adam and J.C. Wortmann 1989]. An evaluation of three data perturbation methods to protect the confidentiality of numerical attributes is presented in [K. Muralidhar and R. Sarathy 1999]. A majority of work in *disclosure limitation* like [Richard A. Moore 1996; Stephen E. Fienberg et al. 1998; Joseph Domingo-Ferrer et al. 2002] focus on applying statistical disclosure limitation methods for categorical and microdatasets. All these methods use techniques like *cell suppression*, *data swapping*, *rounding*, *sampling* and generation of *synthetic data* as a means of achieving statistical disclosure control. While limiting disclosure, they may perturb the characteristics of the original dataset.

Finally, there has been much work on location-based privacy. To the extent of using graph models for privacy analysis, they share some common ground with our work. An example is Beresford and Stajano [Alastair R. Beresford and Frank Stajano 2004]. They consider releasing location information to third-party applications via a middleware mechanism and conduct an analysis of privacy breach that can be achieved by a hacker in this process. However, there are considerable technical differences between the frameworks.

2. BELIEF FUNCTIONS FOR ITEMS AND ITEMSETS

2.1 Anonymization

The original domain is a non-empty universe of items $\mathcal{I}$. For the sequel, we assume $|\mathcal{I}| = n$. A database $\mathcal{D}$ is a sequence of transactions $\langle T_1, \ldots, T_m \rangle$. Each transaction is a non-empty subset of $\mathcal{I}$. As in [Rakesh Agrawal et al. 1993], the frequency of an item $x \in \mathcal{I}$ is the fraction of transactions in $\mathcal{D}$ that contain x.

Let $\mathcal{J}$ be an anonymized domain of items such that $|\mathcal{J}| = |\mathcal{I}|$ and $\mathcal{J} \cap \mathcal{I} = \emptyset$. An *anonymization mapping* is a bijection from $\mathcal{I}$ to $\mathcal{J}$. Transactions are anonymized by replacing each item in the transaction with its anonymized item. Databases are anonymized by anonymizing each transaction. Note that the anonymization mapping is applied uniformly across all the transactions in the database. Hence, if 1 is anonymized to $1'$,this happens in every transaction in the database.

Figure 1 shows a simple example, referred to hereafter as the BigMart example. For the rest of the paper, the primed item, $x' \in \mathcal{J}$, will be used to denote the anonymized item corresponding to $x \in \mathcal{I}$.

Big Mart's Database

TID	Transaction
1	{1,2,3}
2	{1,2,3,4}
3	{4,6}
4	{3,4,5,6}
5	{5,6}
6	{6}
7	{1,2}
8	{1,3,4}
9	{1,3,5}
10	{2,4,6}

I = {1,2,3,4,5,6}

After Anonymization

TID	Transaction
1	{1',2',3'}
2	{1',2',3',4'}
3	{4',6'}
4	{3',4',5',6'}
5	{5',6'}
6	{6'}
7	{1',2'}
8	{1',3',4'}
9	{1',3',5'}
10	{2',4',6'}

J = {1',2',3',4',5',6'}

Fig. 1. Example: Anonymized Database

2.2 Belief Functions for Items

In this paper, we assume that the hacker knows the domain $\mathcal{I}$ and captures this prior knowledge about the domain in a *belief function*, β. A belief function maps each item x in $\mathcal{I}$ to an interval $[l, r]$, modeling the belief that the frequency of x in the database is in the range $[l, r]$ where $0 \leq l \leq r \leq 1$.

Two special belief functions are considered which represent extremes in the hacker's prior belief. When a hacker has *no* knowledge of the frequency of any item in $\mathcal{I}$, each item in $\mathcal{I}$ maps to $[0, 1]$. In this case, the hacker is *ignorant* of the frequency of any item in $\mathcal{I}$ and the belief function is called an *ignorant* belief function.

The other extreme is when the hacker has *exact* knowledge of the frequency of every item in $\mathcal{I}$, with each interval $[l, r]$ essentially becoming a point value in the interval $[0, 1]$. This belief function is called a *point-valued* belief function. A belief

function is called an *interval* belief function if at least one item's belief interval is a true range, i.e., $l < r$.

A belief function is *compliant*, if for every item $x \in \mathcal{I}$, the range $[l, r]$ contains the true frequency of x. A belief function is α-compliant if only a fraction α ($0 \leq \alpha \leq 1$) of items satisfy the requirement of true frequency containment. For simplicity, whenever $\alpha = 1$, we simply refer the belief function as compliant.

Figure 2 shows four belief functions f, g, h and k over Big Mart's domain: f is a compliant point-valued belief function; g is an ignorant (and compliant) belief function; h is a compliant interval belief function; and k is 0.5-compliant, as it guesses wrong on the first three items.

Belief Functions f, g, h, k:

Compliant Point–Valued	*Ignorant*	*Compliant Interval*	*0.5–Compliant Interval*
f(1) = 0.5	g(1) = [0,1]	h(1) = [0,1]	k(1) = [0.1,0.4]
f(2) = 0.4	g(2) = [0,1]	h(2) = [0.4,0.5]	k(2) = 0.5
f(3) = 0.5	g(3) = [0,1]	h(3) = 0.5	k(3) = [0.1,0.3]
f(4) = 0.5	g(4) = [0,1]	h(4) = [0.4,0.6]	k(4) = [0.4,0.6]
f(5) = 0.3	g(5) = [0,1]	h(5) = [0.1,0.4]	k(5) = [0.1,0.4]
f(6) = 0.5	g(6) = [0,1]	h(6) = 0.5	k(6) = 0.5

Fig. 2. Examples of Belief Functions

2.2.1 *Inference and Consistency Assumptions.* A hacker uses a *crack mapping*, $C : \mathcal{J} \rightarrow \mathcal{I}$, to identify anonymized items. We assume the hacker only uses 1-1 crack mappings, i.e., (s)he assigns exactly one item to each anonymized item.

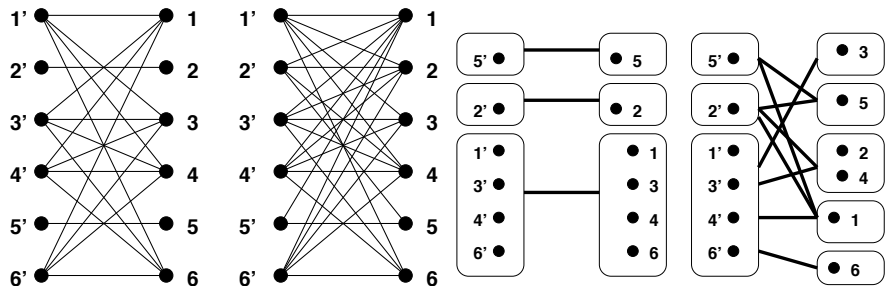
An item $x \in \mathcal{I}$ is said to be *cracked* by C whenever the mapping correctly maps the anonymized item x' to x. The question here is which crack mappings are used by the hacker. We assume that the hacker uses his/her belief function to derive crack mappings. Specifically, let x' be an anonymized item and let $F(x')$ be its observed frequency in the database. Then, x' is mapped by the hacker to only those items in $\mathcal{I}$ whose belief interval contains $F(x')$. Such mappings are called *consistent mappings*, as they are consistent with the prior knowledge a hacker has about $\mathcal{I}$. Henceforth, mapping always means a *consistent* mapping.

Consider for example the belief function h shown in Figure 2. By analyzing the anonymized data, the hacker will surely find out the frequencies of $1'$, $2'$, $3'$, $4'$, $5'$ and $6'$ are respectively 0.5, 0.4, 0.5, 0.5, 0.3 and 0.5 (i.e., specified precisely by the compliant, point-valued belief function f). What can be the true identity of $1'$ then? To be consistent with the belief function h, $1'$ can be mapped to 1, 2, 3, 4 and 6; $h(5) = [0.1, 0.4]$ is the only range not containing 0.5. Similarly, the observed frequency of $2'$ is 0.4, and $2'$ can be mapped to 1, 2, 4 and 5.

Given a belief function and an anonymized database, the space of all consistent crack mappings can be represented by a bipartite graph $G = (\mathcal{J} \cup \mathcal{I}, E)$, where each mapping is a perfect matching in G. The edge (x', y) in G denotes the fact that a mapping used by a hacker can map the anonymized item x' to the item $y \in \mathcal{I}$. Thus, for the ignorant belief function, we have a complete bipartite graph. Note

that, if a belief function is compliant on item x, then the edge (x', x) is present in the bipartite graph.

Figure 3(a) shows the bipartite graphs corresponding to the belief functions f and h in Figure 2. For the latter, as discussed in the previous paragraph, 1' is connected to 1, 2, 3, 4 and 6; 2' is connected to 1, 2, 4 and 5; and so on.



(a) Space of Mappings for Items (b) Space of Mappings for Frequency Groups

Fig. 3. Space of Consistent Mappings for f, h

Figure 3(b) shows an alternative presentation of the mappings of Figure 3(a) by grouping the items in the anonymized domain and the original domain and viewing mappings in terms of these groups. The anonymized items can be grouped based on their observed frequencies. On the other hand, the items in the original domain can be grouped based on the anonymized items that can map to them. Specifically, items x and y in $\mathcal{I}$ belong to the same group if $\{w'|(w', x) \in G\} = \{w'|(w', y) \in G\}$, where G denotes the bipartite graph. Consider the group mapping corresponding to the belief function h (of figure 2). Even though items 2 and 4 have different belief intervals, they belong to the same group as the same set of anonymized items can map to these two items.

Note that given a belief function and a corresponding bipartite graph, there may not exist a perfect matching. As a simple example, let the original domain be {1, 2} and the corresponding anonymized domain be {1', 2'}. The bipartite graph may map both 1' and 2' to 2, with no anonymized item mapped to 1. Throughout our analysis, situations like this are dealt with using α-compliancy (Section 4.2). Until then, we consider only compliant belief functions.

2.3 Belief Functions for Itemsets

So far, we have only focused on belief functions for single items, i.e., singleton itemsets. Below we extend the notion introduced above to itemsets of arbitrary size. There are two issues. First, when dealing with items, we are dealing with a domain $\mathcal{I}$ of items of a reasonable size (say n). However, when we generalize this directly to itemsets, we are suddenly dealing with an exponential number of itemsets (2^n possible itemsets). Even for a reasonably small domain of say 100 items, this is a large number of itemsets to deal with. Second, even if we do manage to compute the expected number of itemsets that are cracked, the data owner has no idea which are the individual itemsets that are more likely or less

likely to be cracked. Suppose the expected number of cracks is 14000, the owner has no idea which 14000 itemsets out of a possible 2^n are likely to be cracked. For the reasons discussed above, instead of dealing with the entire space of 2^n itemsets, the data owner picks Θ, a set of itemsets of interest. The data owner is interested in ensuring that the fraction of itemsets from Θ that are likely to be cracked with probability at least σ is not more than τ. We formally define this requirement as follows and illustrate these parameters with some examples.

Definition 2.1 *Tolerance Requirement for Itemsets.* Let Θ be a subset of the powerset $2^{\mathcal{I}}$ of items. We say that an itemset is *vulnerable*, if its probability of being cracked is $\geq \sigma$; we call σ, which is a real number in [0,1], the crack-danger threshold. The tolerance requirement for itemsets states that the fraction of itemsets in Θ which are vulnerable, must be $\leq \tau$. This is denoted by the predicate $R(\Theta, \sigma, \tau, M)$ which is true if the requirement is satisfied and false otherwise. The measure M stands for heuristic measures or estimates in case the exact probability cannot be computed.

Example 1: The data owner is interested in the set of all pairs of items and is willing to tolerate at most 10% of the pairs being in-danger, Say, the data owner sets the crack-danger threshold to be $\sigma = 50\%$. Then, $\Theta_0 = \{X \subset 2^{\mathcal{I}} : (|X| = 2)\}$ and the thresholds are $\sigma = 0.5$ and $\tau = 0.1$.

Example 2: Knowing that items which occur very frequently are likely to be known by competitors, the data owner may decide to focus on the items that are less obvious. For example, the data owner may be less interested in the top $K\%$ of the most frequent items and their supersets, and may be more keen to protect the identities of the 2-itemsets formed from the remaining items. For our experiments to be shown later, we consider two values of K: (i) $K = 10\%$ and (ii) $K = 20\%$. We call the set of interested itemsets Θ_1 when $K = 10\%$ and Θ_2 when $K = 20\%$. Thus, for Θ_1 (resp., Θ_2) and parameters $\tau = 0.1$ and $\sigma = 0.5$, the owner is interested in ensuring that 90% of the itemsets in Θ_1 (Θ_2) are each cracked with a probability less than 0.5.

These examples present only a small set of possibilities out of what can be specified by the model. One can still explicitly enumerate all the itemsets of interest or specify them succinctly using other means and also specify different thresholds for τ and σ for different classes of subsets of items. One could use constraints to specify such classes (e.g., as in [Raymond T. Ng et al. 1998]). We do not explore these issues further. Henceforth our focus is on the disclosure risk of item sets, unless otherwise specified.

3. THE PROBABILITY OF CRACKING AN ITEMSET - THE EXTREME CASES

To decide whether to release the anonymized data, the data owner needs to assess the risk of disclosure. Given the bipartite graph corresponding to the belief function, there are many possible crack mappings, giving rise to different numbers of cracks. Throughout this paper, we assume that each consistent crack mapping is equally likely. (In practice, the hacker may use additional prior knowledge to favor one crack mapping over another. We consider this beyond the scope of this work.) Hence, the risk of disclosure can be reasonably measured by the expected percentage or number of cracks. We first consider the two extreme cases of prior knowledge and

derive exact formulas for the probability of cracking a k-itemset in each of these cases.

3.1 The Ignorant Belief Function

When the hacker is ignorant of the frequencies of items in $\mathcal{I}$, (s)he can map an anonymized item to *any* item in $\mathcal{I}$ and the space of mappings is a complete bipartite graph. Consider the ignorant belief function g in Figure 2. There are 6 items in the domain $\mathcal{I}$. What is the probability that the itemset $\{1, 2\}$ is cracked? Those crack mappings in which the items 1 and 2 are mapped to the items $1'$ and $2'$ (irrespective of order) are the mappings in which the itemset $\{1, 2\}$ is cracked. The number of such mappings is given by $2 \times 4! = 48$. The total number of crack mappings for the complete bipartite graph is $6! = 720$. Hence the probability that $\{1, 2\}$ is cracked is $\frac{48}{720} = 0.067$. Since the bipartite graph is complete, this is the probability that *any* itemset of size 2 is cracked independently of others. Generalizing this analysis, let us compute the probability of a k-itemset $\{i_1, \ldots, i_k\}$ being cracked out of a domain of n items. Mappings in which the item set $\{i_1, \ldots, i_k\}$ is cracked are precisely those mappings in which the items $i_1, ..., i_k$ are mapped to the items $i'_1, \ldots, i'_k$ (not necessarily in that order. The set $\{i_1, \ldots, i_k\}$ can map to $\{i'_1, \ldots, i'_k\}$ in $k!$ ways and the remaining items can be mapped in $(n-k)!$ ways. There are thus a total of $k! \times (n-k)!$ mappings in which the k-itemset is cracked. The complete bipartite graph on n items admits a total of $n!$ crack mappings and thus the probability of the k-itemset being cracked is $\frac{k! \times (n-k)!}{n!} = \frac{1}{\binom{n}{k}}$. We just proved the following lemma:

LEMMA 3.1. *Let $G = (\mathcal{J} \cup \mathcal{I}, E)$ be a complete bipartite graph modeling the crack space of an ignorant belief function f. The probability of cracking a k-itemset using a consistent mapping from G, is $\frac{1}{\binom{n}{k}}$.*

PROOF. Follows from the arguments above. □

LEMMA 3.2. *Let G be a complete bipartite graph. Let X be a random variable representing the number of k-itemsets that are cracked by a randomly chosen mapping obtained from G. Then the expected number of cracks, $E(X) = 1$.*

PROOF. Follows from Lemma 3.1 and the observation that there are $\binom{n}{k}$ itemsets of size k. □

As a result of the lemma, the expected *fraction* of cracks $\frac{1}{\binom{n}{k}}$ is small for large domains and thus, for large domains, when the hacker is ignorant,[4] anonymization is indeed a good option for the data owner. What can we say about the expected number of cracks, when only a subset of k-itemsets are of interest to the data owner? We can consider this question in two ways, depending on how the itemsets of interest are specified: (1) "all k-itemsets chosen from a subset of items $I_1 \subset I$ are of interest" or (2) the enumerated set of k-itemsets $S_1, S_2, ..., S_\ell$ are of interest. The above lemma can be generalized to yield the expected number of cracks as follows.

[4]In practice, the hacker is seldom ignorant. The main message of this analysis is that cracking an itemset by chance is very unlikely for large domains.

LEMMA 3.3. *Let G be a complete bipartite graph. (1) Let $\mathcal{I}_1 \subset \mathcal{I}$ be a subset of items of interest and let $|\mathcal{I}_1| = n_1$. Let X be a random variable representing the number of interested k-itemsets (that are subsets of I_1) that are cracked in a mapping obtained from G. Then the expected number of interested k-itemsets cracked is $E(X) = \frac{\binom{n_1}{k}}{\binom{n}{k}}$. (2) Let $S_1, S_2, ..., S_\ell$ be the k-itemsets of interest. Then the expected number of k-itemsets of interest that are cracked is $E(X) = \frac{\ell}{\binom{n}{k}}$.*

PROOF. Follows from the fact that any k-itemset has a probability of $\frac{1}{\binom{n}{k}}$ of being cracked. In (1) the number of interested k-itemsets is $\binom{n_1}{k}$ while in (2) this number is ℓ. □

3.2 Compliant Point-Valued Belief Function

A *paranoid* data owner may analyze risk assuming that the hacker has exact knowledge of every item's frequency. Note that this extreme case may be unrealistically conservative. Nonetheless, the analysis leads to interesting observations. The compliant point-valued belief function which models this exact knowledge, partitions the bipartite graph of crack mappings into components based on the frequency of the items. Consider the compliant point-valued belief function f shown in Figure 2 and the bipartite graph representing its crack space shown in Figure 3(a). Let us compute the probability of cracking the itemset $\{2, 3, 4\}$. Note that items 3 and 4 have the same frequency and hence lie within the same frequency group. The item 2 has a distinct frequency and is in a singleton frequency group. Hence the item 2 is cracked perfectly (with probability 1). However, the items 3 and 4 lie in the group consisting of the items $\{1, 3, 4, 5\}$ and within this group, the hacker has no further knowledge to distinguish between the items. Also, the cracking of items 3 and 4 is independent of cracking the item 2 as they lie in different frequency groups. By Lemma 3.1, the probability of cracking the itemset $\{3, 4\}$ in the group $\{1, 3, 4, 5\}$ is $\frac{1}{\binom{4}{2}} = \frac{1}{6}$. Hence the probability of cracking the itemset $\{2, 3, 4\}$ is the product $1 \times \frac{1}{6} = \frac{1}{6}$. This result is generalized and formally stated as follows.

LEMMA 3.4. *Let $G = (\mathcal{I} \cup \mathcal{J}, E)$ be a bipartite graph representing the crack space of a compliant point-valued belief function. Let $G_1, \ldots, G_c$ be the items in $\mathcal{I}$ in the various connected components of G and let $|G_i| = m_i$, for $1 \leq i \leq c$. Let $\mathcal{I}_k \subset \mathcal{I}$ be a k-itemset and let $X_i = \mathcal{I}_k \cap G_i$ and $|X_i| = x_i$, for $1 \leq i \leq c$. Then, the probability of cracking the itemset $\mathcal{I}_k$ using a consistent mapping from G is $\prod_{i=1}^{c} \frac{1}{\binom{m_i}{x_i}}$.*

PROOF. The items in $\mathcal{I}_k$ that lie in different frequency groups of G are given by the sets $X_1 = \mathcal{I}_k \cap G_1$, ..., $X_k = \mathcal{I}_k \cap G_k$ respectively. Cracking items within each X_i is independent of cracking items in X_j, $j \neq i$. Hence, the probability that $\mathcal{I}_k$ is cracked is the product of the probability of cracking each of the X_i. Since there are m_i items in the group G_i and each component is a complete bipartite graph, the probability of cracking the items in X_i is $\frac{1}{\binom{m_i}{x_i}}$, and multiplying the probabilities across the groups proves the claim. □

Let us next consider the situation where there are g distinct observed frequencies for the anonymized items, thus splitting the graph G into g connected components

$G_1, ..., G_g$. Note that each G_i is a complete bipartite graph. What can we say about the expected number of k-itemsets that can be cracked using a mapping obtained from G? For simplicity, let us assume $|G_i| \geq k$ and also that $k \leq g$.

LEMMA 3.5. *Let g be the number of distinct observed frequencies of the anonymized items. Let G be a bipartite graph modeling the crack space of the compliant point-valued belief function. Let X be a random variable denoting the number of cracks in a mapping obtained from G. Then the expected number of cracks, $E(X) = \sum_{1 \leq i \leq k} \binom{g}{i}\binom{k-1}{i-1}$.*

PROOF. Let S be a k-itemset. In general, S is partitioned into $S \cap G_1, S \cap G_2, ..., S \cap G_g$. Let $m_i = |G_i|$ and $x_i = |S \cap G_i|$. Then the probability that S is cracked is the probability that each of the sets above is cracked. Cracking of itemsets in different components is clearly independent of each other. The probability of $S \cap G_i$ (whose cardinality is x_i) being cracked is $\frac{1}{\binom{m_i}{x_i}}$. So, the probability of S being cracked is $\frac{1}{\binom{m_1}{x_1} \times \cdots \times \binom{m_g}{x_g}}$. Notice that itemset S' such that $|S' \cap G_i| = x_i$, $1 \leq i \leq g$ has the same probability of being cracked as S. There are precisely $\binom{m_1}{x_1} \times \cdots \times \binom{m_g}{x_g}$ such itemsets, so the expected number of k-itemsets that may be cracked, among those that are split between the g components exactly like S, is 1. The number of k-itemsets that have a non-empty overlap with i components is the number of ordered partitions of k into i parts, and is $\binom{k-1}{i-1}$. The total expected number of cracks of k-itemsets is thus $\binom{g}{1} + \binom{g}{2}\binom{k-1}{1} + \binom{g}{3}\binom{k-1}{2} + \cdots + \binom{g}{i}\binom{k-1}{i-1} + \cdots + \binom{g}{k}\binom{k-1}{k-1}$ $= \sum_{1 \leq i \leq k} \binom{g}{i}\binom{k-1}{i-1}$. □

To get a feel for what this means, let us consider an example. Let $n = 100$, $g = 4$, $m_1 = \cdots = m_4 = 25$ and $k = 3$. The total number of k-itemsets in the domain is $\binom{100}{3}$. The expected number of k-itemsets cracked is $4 + \binom{4}{2}\binom{2}{1} + \binom{4}{3}\binom{2}{2} = 20$, which is a very small fraction of $\binom{100}{3}$. The analysis above easily extends to the case where $|G_i| < k$ for some components, or $k > g$, or both. The details are omitted.

Note that even if the hacker has exact knowledge, the expected number of cracks need not necessarily be $\binom{n}{k}$, as the items with equal frequencies provide camouflage to each other so that their true identities are protected. The above lemma can be generalized to handle a subset $\mathcal{I}_1 \subset \mathcal{I}$ of items of interest by using Lemma 3.3 on each frequency group independently.

LEMMA 3.6. *Let g be the number of distinct observed frequencies of the anonymized items. Let $m_1, \ldots, m_g$ be the size of each of the frequency groups. Let $c_1, \ldots, c_g$ be the number of items the data owner is interested in, for each of the frequency groups. Let $G = (\mathcal{J} \cup \mathcal{I}, E)$ be the bipartite graph representing the space of all mappings and let X be a random variable representing the number of interested k-itemsets cracked by a mapping obtained from G. Then, the expected number of interested k-itemsets being cracked is $E(X) = \sum_{i=1}^{g} \binom{g}{i} \times \sum_{x_1 + \cdots x_i = k \ \&\ x_j \neq 0, 1 \leq j \leq i} \frac{\binom{c_1}{x_1} \cdots \binom{c_i}{x_i}}{\binom{m_1}{x_1} \cdots \binom{m_i}{x_i}}$.*

PROOF. The main difference with the proof of Lemma 3.5 is that in Lemma 3.5, every collection of k-itemsets with an overlap of x_1 items from G_1, x_2 items from G_2, ..., x_i items from G_i had an expected number of cracks of 1. In the current

lemma, however, this expectation drops to $\frac{\binom{c_1}{x_1}\cdots\binom{c_i}{x_i}}{\binom{m_1}{x_1}\cdots\binom{m_i}{x_i}}$. Thus instead of the factor $\binom{g}{i}$, we must instead use the weighted sum $\sum_{x_1+\cdots x_i=k \ \&\ x_j\neq 0, 1\leq j\leq i} \frac{\binom{c_1}{x_1}\cdots\binom{c_i}{x_i}}{\binom{m_1}{x_1}\cdots\binom{m_i}{x_i}}$. The remaining steps of the reasoning are identical and are omitted. □

4. COMPLIANT INTERVAL BELIEF FUNCTIONS

The analysis of risk for extreme cases is not very realistic in practice. From the data owner's perspective, the risk analysis for exact knowledge is the worst case. In this section, we analyze the more realistic situation when the hacker's belief is a compliant interval belief function, where the hacker associates arbitrary intervals tighter than $[0, 1]$ with various items in the domain. The belief function h in figure 2 is an example of an interval belief function.

4.1 The O-Estimate for Items

Let us begin with the simpler case of dealing with singleton itemsets, i.e., $k = 1$. Even for this simplification, a direct method to compute the expected number of cracks for an arbitrary compliant interval belief function involves computing permanents of the bipartite graph representing the crack space and its many induced subgraphs. For more details of the exact formula, the reader is referred to [Laks V.S. Lakshmanan et al. 2005]. The problem is that computing the permanent is known to be very difficult – more precisely, a $\#P$-complete problem [Leslie G. Valiant 1979]. Various approximations have been developed for computing permanents [Mark Jerrum and Umesh Vazirani 1996; L.E. Rasmussen 1994]. The state of the art is the polynomial-time randomized approximation scheme presented in [Mark Jerrum et al. 2001]. However, the running time is of the order of $O(n^{22})$! Thus, for arbitrary interval belief functions and realistic domain sizes, it is not feasible to use the direct approach to compute the expected number of cracks. Below we provide a heuristic algorithm that applies to a general interval belief function and computes an approximation to the expected number of cracks.

Note that the proof of the previous lemmas on singleton itemsets repeatedly apply a well-known result from statistics. Let X and Y be two random variables. Then it is the case that $E(aX + bY) = aE(X) + bE(Y)$. This identity does not require that X and Y be independent. Thus, for a general bipartite graph, we can still analyze the expected number of cracks by examining each item in an isolated fashion.

Let β be an interval belief function and let G be the bipartite graph representing the space of all mappings. For each $x \in \mathcal{I}$, let O_x denote the outdegree of the node x in G. The outdegree of x basically denotes the number of anonymized items that can be mapped to it. Note that the probability that the anonymized item x' correctly maps to x is given by $\frac{1}{O_x}$ (under the compliancy assumption, this edge is always guaranteed to exist in the space of mappings). Thus, the expected number of cracks can be estimated by summing this probability across all the items in $\mathcal{I}$. We call this heuristic the **O-estimate** ("O" standing for "Outdegree") (denoted $OE(\beta, \mathcal{D})$) and it is defined as $\sum_{x\in\mathcal{I}} \frac{1}{O_x}$. Figure 4 outlines a procedure to compute the O-estimate, given a belief function and a database. A little analysis shows that the total running time of an efficient implementation is $O(|\mathcal{D}| + n \log n)$ (for details

Algorithm O-estimate
Input: β - interval belief function over $\mathcal{I}$, $\mathcal{D}$ - Anonymized Database over $\mathcal{J}$
1. Compute frequency of single items in $\mathcal{J}$ from $\mathcal{D}$.
2. Compute the frequency groups $g_1, \ldots, g_k$.
   Let $n_1, \ldots, n_k$ be their sizes and $f_1 < f_2 \cdots < f_k$ be their frequencies.
3. Initialize $O_{est} = 0$
4. For each $x \in \mathcal{I}$
   a. Compute n_x, the number of anonymized items that can map to x.
   b. $O_{est} = O_{est} + \frac{1}{n_x}$
5. return O_{est};

Fig. 4. The O-estimate Heuristic

Algorithm Propagate
Input: G - A fully compliant bipartite graph over $\mathcal{I} \cup \mathcal{J}$
1. while there is a node with degree 1 in G
   Let x be this node (the same holds for x').
   a. Remove the nodes x and x' from G and their incident edges.
   b. For each y such that the edge (x', y) is just removed, decrement O_y by 1.

Fig. 5. Reducing the Outdegrees by Propagation

see [Laks V.S. Lakshmanan et al. 2005]).

Let us try to understand why the O-estimate is not exact. Consider the example in Figure 6(a). The outdegrees of $1, 2, 3$ and 4 are $1, 2, 3$ and 4 respectively and the O-estimate is $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} = \frac{25}{12}$. However, consider node 1. Only $1'$ can map to it in any perfect matching. This leads to the other edges on 1 to be removed from the graph. As a consequence, $2'$ is the only node that can map to 2 and this cascades to the scenario when $1'$, $2'$, $3'$ and $4'$ map to 1,2,3 and 4 respectively. Hence, the number of cracks is 4. The same argument can be given by starting with node $4'$ (which is also of degree 1).

As a result, when the outdegree of a node is 1, propagation may reduce the outdegrees of other items. Figure 5 outlines an $O(ve)$ procedure to handle this propagation, where v is the number of nodes in the graph and e is the number of edges in the graph. Even though the propagation may go on for v steps in the worst case, the fixed point is often reached in much fewer iterations in practice. This propagation procedure should be applied after step $4(a)$ in Figure 4. Hereafter, we assume that propagation has been done before computing outdegrees.

The discussion so far focuses on a special case when an (anonymized) item has an outdegree 1. A similar phenomenon occurs when the outdegree is 2 or higher. The example in Figure 6(b) differs from that in Figure 6(a) in that no item can be surely cracked. However, no perfect matching can contain the edge $\{2', 3\}$ and this edge is hence irrelevant. Yet, the O-estimate counts the edge for the outdegree of item 3.

In spite of the shortcomings discussed so far, O-estimates have a nice property, namely, *monotonicity*. As the belief interval for an item x gets wider, the number of anonymized items that can map to it increases and hence, the expected number of cracks drops.

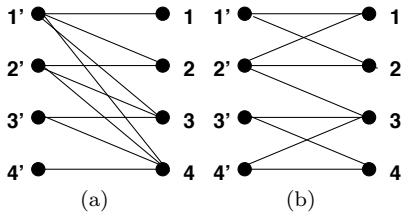


Fig. 6. Inexactness of the O-estimates

Definition 4.1. Let β_1 and β_2 be two interval belief functions on $\mathcal{I}$. Then, $\beta_1 \preceq \beta_2$ whenever $\forall x \in \mathcal{I}$, $\beta_1(x) \subseteq \beta_2(x)$. We say that an interval $[l_1, r_1] \subseteq [l_2, r_2]$ whenever $l_1 \geq l_2$ and $r_1 \leq r_2$.

As the uncertainty in belief (width of intervals) grows, we would expect it to be more difficult on the average to crack individual items as captured by the following lemma. We will see in the next section how this property can be made use of.

Lemma 4.2 Monotonicity of OE. *Let $\mathcal{D}$ be an anonymized database. Let β_1 and β_2 be two compliant interval belief functions such that $\beta_1 \preceq \beta_2$. Then, $OE(\beta_1, \mathcal{D}) \geq OE(\beta_2, \mathcal{D})$.* □

4.2 α-Compliant Belief Functions for Items

All our derivations so far have assumed *full compliancy*, i.e., for each item, the hacker's belief interval contains the true frequency of the item. This is possible if the hacker has very good knowledge about the items, or if the intervals are wide or conservative. In practice, there is no reason to believe full compliancy is always possible, or even likely and hence, we examine α-compliant belief functions. However, they are tricky to deal with because the hacker would have no idea which ones of his/her guessed intervals are not correct. (Had (s)he known, (s)he would have changed them in the first place!) We defer this issue to the next section and proceed with the analysis of computing the expected number of cracks, assuming that we somehow *know which items are guessed wrong*.

Let β be a belief function on $\mathcal{I}$ such that for $\mathcal{I}_C \subset \mathcal{I}$, the function is compliant and for $\mathcal{I} - \mathcal{I}_C$ the function is non-compliant. Thus, α is defined as the ratio of $|\mathcal{I}_C|$ to that of $|\mathcal{I}|$. We apply the O-estimate to approximate the expected number of cracks. However, the consistency assumption guarantees that the items $x \in (\mathcal{I} - \mathcal{I}_C)$ will not be cracked. Thus, the O-estimate for β simply sums over those items $x \in \mathcal{I}_C$ and is defined as $OE(\beta, \mathcal{D}) = \sum_{x \in \mathcal{I}_C} \frac{1}{O_x}$.

Intuitively, when belief functions become more and more non-compliant, the expected number of cracks should decrease, as non-compliant items cannot be cracked by a consistent mapping. The following lemma formalizes this monotonicity.

Definition 4.3. Let β_1 and β_2 be two interval belief functions on $\mathcal{I}$. Let β_1 be compliant on the set of items $\mathcal{I}_C^1 \subset \mathcal{I}$, and β_2 be compliant on $\mathcal{I}_C^2 \subset \mathcal{I}$. We say that $\beta_2 \preceq_C \beta_1$ whenever: (i) $\mathcal{I}_C^2 \subseteq \mathcal{I}_C^1$; and (ii) $\forall x \in \mathcal{I}_C^2$, $\beta_1(x) \subseteq \beta_2(x)$.

The above definition imposes a partial order on α-compliant belief functions, based on the subset of compliant items. As this subset becomes smaller and smaller,

the expected number of cracks becomes smaller as the guessed intervals of the compliant items do not shrink.

LEMMA 4.4 MONOTONICITY OF OE FOR α-COMPLIANCY. *Let $\mathcal{D}$ be an anonymized database. Let β_1 and β_2 be two interval belief functions such that $\beta_2 \preceq_C \beta_1$. Then, $OE(\beta_2, \mathcal{D}) \leq OE(\beta_1, \mathcal{D})$.* □

4.3 The OS Estimate for Itemsets

The cracking of different items in a set is usually dependent on each other. The O-estimate heuristic for items basically relaxes this assumption and computes the probability of cracking an item independent of others and uses this to compute the expected number of cracks. We use a similar technique and propose two heuristics for estimating the probability of cracking itemsets.

Let G be the bipartite graph representing the space of crack mappings for an interval belief function β. Let $\mathcal{I}_k = \{i_1, i_2, \ldots, i_k\}$ be a k-itemset. Let $N(i_j)$ denote the set of neighbors of i_j in G. A consistent mapping in G can thus map i_j to any of the anonymized items in $N(i_j)$. The itemset $\mathcal{I}_k$ is cracked whenever each item in $\mathcal{I}_k$ maps to any of the anonymized items $\mathcal{I}'_k = \{i'_1, \ldots, i'_k\}$. Let us first consider the item i_1. The probability that item i_j maps to any of the items in $\mathcal{I}'_k$ (independent of others) is given by $\frac{|N(i_j) \cap \mathcal{I}'_k|}{|N(i_j)|}$. By assuming the independence of events, the probability that the items in $\mathcal{I}_k$ map to the items in $\mathcal{I}'_k$ is given by $\prod_{j=1}^{k} \frac{|N(i_j) \cap \mathcal{I}'_k|}{|N(i_j)|}$ and we call this estimate the OS heuristic ("OS" standing for "Outdegree for Sets"). Whenever $N(i_j)$ is empty for even a single i_j, the resulting estimate is assumed to have a value of 0. More formally,

$$
\begin{aligned}
OS(\mathcal{I}_k, \beta) &= 0\,, && \text{if } \exists x \in \mathcal{I}_k : N(x) = \emptyset \\
&= \prod_{x \in \mathcal{I}_k} \frac{|N(x) \cap \mathcal{I}'_k|}{|N(x)|}, && \text{otherwise}
\end{aligned}
$$

We have considered a second way to estimate the probability of cracking $\mathcal{I}_k$ by treating the entire set of items as a single entity. In doing so, we merge all the nodes corresponding to the items in $\mathcal{I}_k$ and take the union of the neighbors of the items in $\mathcal{I}_k$. If the union contains only a part of $\mathcal{I}'_k$, then the probability of cracking the itemset $\mathcal{I}_k$ is zero. Otherwise, we merge the items corresponding to $\mathcal{I}'_k$ in the union and compute the probability as the ratio of 1 over the size of this union. This estimate is called the $OS2$ heuristic and is formally defined as

$$
\begin{aligned}
OS2(\mathcal{I}_k, \beta) &= 0\,, && \text{if } \mathcal{I}'_k \not\subseteq \bigcup_{x \in \mathcal{I}_k} N(x) \\
&= \frac{1}{|\bigcup_{x \in \mathcal{I}_k} N(x)| - k + 1}, && \text{otherwise}
\end{aligned}
$$

However, in all our experiments, OS almost always dominates $OS2$ and hence, we drop further discussion on the $OS2$ estimate.

Note that the definition of OS directly applies to α-compliant belief functions, under the same assumptions as discussed in Section 4.2. Figure 7 outlines a procedure to compute the OS estimate for a single itemset X. Step 1 makes one pass

Algorithm Compute-OS-Estimate
Input: X - An itemset over $\mathcal{I}$, β - Belief Function, $\mathcal{D}$ - Anonymized Database over $\mathcal{J}$
1. For each $x' \in \mathcal{J}$, Compute the frequency $F(x')$ in $\mathcal{D}$.
2. $OS = 1$
3. For each $x \in X$
   a. Compute $n_x =\mid N(x) \mid$.
   b. $i_x = 0$
   c. For each $x' \in X'$
      if $F(x') \in \beta(x)$, increment i_x
   d. $OS = OS \times \frac{i_x}{n_x}$
4. return OS.

Fig. 7. Computing the OS Estimate

over the database $\mathcal{D}$ and uses $O(n)$ space, where $|\mathcal{I}| = |\mathcal{J}| = n$. Step 3 runs for $|X|$ iterations. Step $3(a)$ takes $O(n)$ time and $O(n)$ space. Step $3(c)$ takes $O(|X|)$ time. Thus, step 3 in total takes $O(n \times |X|)$ time and $O(n)$ space.

Let us illustrate the OS estimate with an example. Consider the crack space shown in Figure 6(b). There are four possible consistent mappings for this crack space. The probability of cracking itemset $\{1, 2\}$ is 1 as every consistent mapping maps the items 1 and 2 to the items $1'$ and $2'$. According to the OS heuristic, the probability of 1 mapping to the items $1'$ and $2'$ is 1 as $N(1) = \{1', 2'\}$. The same goes for 2 and hence the OS heuristic for $\{1, 2\}$ gives a value 1.

Consider the itemset $\{1, 3\}$. The actual probability of cracking this itemset is $\frac{1}{4}$. However, the probability of 1 mapping to the itemset $\{1', 3'\}$ is $\frac{1}{2}$ and the probability of mapping 3 to the itemset $\{1', 3'\}$ is $\frac{1}{3}$ and hence $OS(\{1, 2\}) = \frac{1}{2} \times \frac{1}{3} = \frac{1}{6}$. Similarly, it can be seen that for the itemset $\{2, 3\}$, the exact probability of cracking this itemset is $\frac{1}{4}$ while $OS(\{2, 3\}) = \frac{1}{3}$. Note that while the OS heuristic underestimates the probability of cracking $\{1, 3\}$, it overestimates the probability of cracking $\{2, 3\}$. The discrepancy is due to the assumption that cracking these itemsets are independent. We will study its behavior in detail in the experimental section.

The OS estimate has the nice property of *monotonicity* which is formalized for compliant and α-compliant interval belief functions below. For the following results $\preceq$ and $\preceq_C$ are given by Definitions 4.1 and 4.3 respectively.

LEMMA 4.5 MONOTONICITY OF OS. *Let $\mathcal{D}$ be an anonymized database and let β_1 and β_2 be two interval belief functions such that $\beta_2 \preceq \beta_1$. Then, for an itemset $X \subseteq \mathcal{I}$, $OS(X, \beta_1) \geq OS(X, \beta_2)$.*

PROOF. Let G_1 and G_2 be the bipartite graph representing the crack spaces for β_1 and β_2 respectively. Then, for each $x \in X$, $N_1(x) \subseteq N_2(x)$ where N_1 and N_2 are the neighbors in graphs G_1 and G_2 respectively. The proof essentially follows from this observation and the definition of OS. □

The proof of monotonicity for α-compliancy is similar and follows from the definition of $\preceq_C$.

LEMMA 4.6 MONOTONICITY FOR α-COMPLIANCY. *Let $\mathcal{D}$ be an anonymized database*

and let β_1 and β_2 be two interval belief functions such that $\beta_2 \preceq_C \beta_1$. Then, for an itemset $X \subseteq \mathcal{I}$, $OS(X, \beta_1) \geq OS(X, \beta_2)$. □

5. RECIPES FOR RISK ASSESSMENT

Thus far, we studied various belief functions as a means of capturing information that a hacker may possess. We now tackle the original dilemma facing the data owner, of whether to release the anonymized data: *Just how safe is the anonymized data in the presence of partial information?*

5.1 A Recipe for Items

Let us begin with the absolute worst case: the compliant, point-valued belief function. We feel that for most applications, the worst case is too conservative, as it is unrealistic to expect the hacker to know each frequency precisely. The expected percentage of cracks is typically unrealistically high in this case. Thus, the owner is ill advised to decide based on this value, unless (s)he is paranoid. It makes sense to relax the worst case in the following two ways. First, the compliant point-valued belief function can be extended to a compliant interval belief function, i.e., if f_x denotes the true frequency of item x, then the interval guessed by the compliant interval belief function is set to $[f_x - \delta, f_x + \delta]$. For the data owner, this corresponds to the hacker being accurate in guessing a right "ball-park" frequency range for every item. The question here is what is an appropriate value of δ to use. As a heuristic, we propose using the median frequency gap for every item. That is, the differences between two successive frequency groups in the data are computed, and the median of these differences is used.

While more details of experiments will be given in the next section, the table in Figure 10 shows various statistics of the benchmark datasets: the number of items in the domain, the number of transactions, the number of distinct frequency groups, the number of singleton frequency groups, and the mean, median, minimum and maximum frequency gap between successive groups. We make two observations.

—These datasets are chosen to represent various characteristics. For instance, the 130 items of the CONNECT dataset form 125 distinct frequency groups, 122 of which consists of a single item. In contrast, the 7120 items of the PUMSB items cluster into 651 frequency groups. Nevertheless, the large number of singleton frequency groups confirm that for real datasets, the compliant point-valued belief function gives too high an estimate on the percentage of cracks.

—For all the datasets, the median frequency gap is much closer to the minimum than to the maximum. In contrast, the average frequency gap is much larger than the median. Thus, by choosing the median value as the width δ of the intervals (hereafter denoted as δ_{med}), the data owner errs on the conservative side (by monotonicity Lemma 4.2). Thus, as compared with using the median, using the average may under-estimate the percentage of cracks. Other alternatives for the width of intervals are discussed in Section 6.2.3.

Figure 8 shows a recipe we suggest to a data owner to assess the risk of releasing anonymized data. Steps (1) to (7) follow what we have discussed so far. Notice that the recipe requires an input percentage τ, called the *degree of tolerance*, which

Algorithm Assess-Risk
Inputs: τ - degree of tolerance; $\mathcal{D}$ - Anonymized Database
1. Compute g based on Lemma 3.5.
2. If $g \leq (\tau \times |\mathcal{I}|)$, disclose $\mathcal{D}$ and stop.
3. Compute the frequency groups from $\mathcal{D}$, and the median gap M between frequency groups
4. Set width δ_{med} to be M.
5. Set up $\beta(x) = [f_x - \delta_{med}, f_x + \delta_{med}]$ for $x \in \mathcal{I}$, where f_x denotes the frequency of $x \in \mathcal{I}$.
6. Compute the O-estimate $OE(\beta, \mathcal{D})$ according to Figure 4.
7. If $OE(\beta, \mathcal{D}) \leq (\tau \times |\mathcal{I}|)$, disclose $\mathcal{D}$ and stop.
8. Set α to be 1.
9. Perform a binary search on α to determine the largest α so that the corresponding α-compliant belief function β satisfies the condition $OE(\beta, \mathcal{D}) \leq (\tau \times |\mathcal{I}|)$.
10. Return the value of α.

Fig. 8. A Suggested Recipe for Risk Assessment

gives a fraction of the items $\mathcal{I}$ that the data owner can tolerate being cracked. If the expected number of cracks based on the compliant point-value belief function is already within the tolerance, then it is an easy decision to release the anonymized data, and the algorithm stops in step (7). In a more likely case, however, this estimate is too high. The recipe then suggests computing the O-estimate based on the compliant interval belief function with the width δ_{med} (i.e., steps (3) to (7)). Note that we use a uniform width of $2 \times \delta_{med}$ for all the intervals. This does not restrict a hacker from using a non-uniform width as (s)he might have varying levels of knowledge about different items. But even if this is true, there is little reason to believe that the data owner has access to the hacker's belief function. Thus, in the recipe, it is a reasonable simplification to use a uniform width.

It is possible that the O-estimate based on the compliant interval belief function is still higher than the owner's tolerance. After all, it is unlikely that the belief function is fully compliant, particularly if the domain is large. Thus, we resort to a second possible relaxation of the worst case, namely, α-compliant belief functions.

Applying the O-estimate for an α-compliant β, is tricky in practice. The difficulties are that (i) it is not clear which values of α could be used, and (ii) even if an α value is established, which specific subset of compliant items $\mathcal{I}_C$ should we use.

Instead of picking specific α values to use, our approach is to examine the risk over a range of α values. Specifically, we use the data owner's specified tolerance τ to determine the largest α value for which the O-estimate falls within the tolerance. Given the monotonicity stated in Lemma 4.4, we can use a binary search to find this value α_{max}. Essentially, it says that in order for the hacker not to crack more than the fraction τ of items that the owner can tolerate, the hacker must not correctly guess the frequency intervals of more than $\alpha_{max} \times |\mathcal{I}|$ items. It is up to the owner to decide whether α_{max} is high enough for comfort. For example, if $\alpha_{max} = 0.8$, then the data owner may decide to disclose the data because the owner considers it highly unlikely that the hacker can guess correctly the frequency intervals, within δ_{med}, of 80% of the items, particularly when the domain is large. On the other hand, if $\alpha_{max} = 0.2$, the data owner may decide to withhold the data because 20% correct guesses may be too small for comfort. Later in Figure 15, a heuristic is proposed to evaluate whether α_{max} is high enough.

Steps (8) and (9) in Figure 8 outline the binary search. There is, however,

one important detail. The binary search is based on the monotonicity stated in Lemma 4.4, which requires a partial ordering on the subset of compliant items $\mathcal{I}_C$. This brings us back to the general issue of how to determine membership in $\mathcal{I}_C$. Our approach is to take a random selection of items and average over a few runs. For example, let us say that when $\alpha = 1$, step (7) gives too high an expected number of cracks. Thus, the algorithm enters into the first iteration of the binary search, with $\alpha = 0.5$. Say the algorithm averages over 5 runs. Thus, there are five subsets of compliant items $\mathcal{I}_C^1, \ldots, \mathcal{I}_C^5$, each having 50% of items randomly picked to be non-compliant. These subsets give rise to five 0.5-compliant belief functions $\beta_1, \ldots, \beta_5$. Thus, the average value of $OE(\beta_1, \mathcal{D}), \ldots, OE(\beta_5, \mathcal{D})$ is used in the condition check in Step (9). Let us suppose that the above average value is still beyond the owner's tolerance, and the binary search continues with $\alpha = 0.25$. In that case, half of the compliant items in each of $\mathcal{I}_C^1, \ldots, \mathcal{I}_C^5$ are randomly picked to be non-compliant. The search then continues as discussed before. Anchoring step (9) in this manner over multiple $\mathcal{I}_C$'s satisfies the requirement of Lemma 4.4.

5.2 A Recipe for Itemsets

Recall that for the case of itemsets, the data owner provides three parameters: Θ, a set of itemsets that the owner wishes to protect, σ, a probability threshold and τ, a fraction of itemsets in Θ for which a breach can be tolerated. This is modeled as the tolerance requirement given in Definition 2.1. The recipe for risk analysis for itemsets is very similar to that for items except that we now analyze the fraction of Θ that are cracked with a probability of at least σ, for the various belief functions. The data owner may start risk analysis with the absolute worst case and analyze the risk when the hacker has exact knowledge, namely, the compliant point-valued belief function. We feel that this case is too conservative in practice and since many items are cracked perfectly under this assumption, the itemsets are also likely to be cracked with a very high probability (due to the presence of a large number of singleton frequency groups and groups of small cardinality). Hence, it makes sense to relax the worst case and this relaxation is done by relaxing the belief function exactly as in Section 5.1.

Algorithm Assess-Risk-Itemsets
Inputs: σ - Probability Threshold; $\mathcal{D}$ - Anonymized Database, τ - Fraction of Itemsets;
Θ - Itemsets of Interest
1. For each $X \in \Theta$, Compute exactly, the probability of cracking X using Lemma 3.4.
2. If this satisfies the tolerance requirement of Definition 2.1, disclose $\mathcal{D}$ and stop.
3. Compute the frequency groups from $\mathcal{D}$, and the median gap M between frequency groups
4. Set width δ_{med} to be M.
5. Set up $\beta(x) = [f_x - \delta_{med}, f_x + \delta_{med}]$ for $x \in \mathcal{I}$, where f_x denotes the frequency of $x \in \mathcal{I}$.
6. Compute OS estimate for each $X \in \Theta$.
7. If the tolerance requirement of Definition 2.1, $R(\Theta, \sigma, \tau, OS)$ is true, disclose $\mathcal{D}$ and stop.
8. Set α to be 1.
9. Perform a binary search on α to determine the largest α so that the corresponding α-compliant belief function β satisfies the tolerance requirement $R(\Theta, \sigma, \tau, OS)$.
10. Return the value of α.

Fig. 9. A Recipe for Risk Assessment for Itemsets

Figure 9 shows a new recipe that we suggest to the data owner for disclosing anonymized data when risk analysis is done for itemsets. Note that the recipe is essentially the same as that for items in terms of considering the various belief

functions and their compliancy. The selection of the width of intervals and determining the degrees of compliancy are exactly as in Sections 5.1 and 4.3. The one change is the estimate that is computed and its interpretation. Instead of computing the O-estimate for the expected number of cracks, the recipe computes the fraction of itemsets in Θ that are cracked with a probability at least σ and this fraction is compared with the owner's tolerance threshold τ. This is first computed for the compliant interval-valued belief function – the data owner assumes that the hacker has a compliant interval belief function. If the fraction is too high, then the tolerance requirement is not satisfied. In many cases, the assumption that the hacker's belief function is fully compliant may also turn out to be too conservative. The recipe thus computes the estimates for an α-compliant belief function. Instead of picking a specific value for α, the risk is analyzed over a range of compliancy values and the largest compliancy value α_{max}, for which the tolerance requirement is satisfied is returned to the data owner. The value α_{max} thus models the smallest fraction of items on which the hacker has to be compliant to breach the tolerance requirement. Like before, instead of explicitly making a decision about disclosure, it is for the owner to decide whether α_{max} is high enough for comfort.

6. EXPERIMENTAL EVALUATION

In this section, we empirically evaluate the accuracy of the various estimates and the effectiveness of the recipes.

6.1 Experimental Setup

The table in Figure 10 shows the characteristics of the benchmark datasets obtained from [S. Hettich and S.D. Bay 1999; FIMI 2003] used in our experiments. The domain varies from 75 to 16470 items, and the number of transactions varies from 3196 to 340184. In many datasets, the number of singleton frequency groups is high in relation to the total number of items confirming that compliant point-valued belief functions are too conservative in practice. However, the RETAIL dataset is very different. The ratio of the number of transactions to the domain size is 1-2 orders of magnitude smaller than those for the other datasets. We label this dataset as "sparse". The table also shows the average, median, minimum and maximum gap between frequency groups.

All procedures were implemented in C++. The code to simulate the expected number of cracks for a given interval belief function needs to generate many consistent matchings from the bipartite graph that represents the crack space. This generation of samples is complicated by the fact that we need matchings that are perfect, consistent, and as much as possible, random. To do so, the generation procedure initially starts with a perfect matching where every edge is of the form (i', i) i.e, every item is cracked. Then, for a fixed number of iterations $(100,000)$, a random permutation P of $\mathcal{I}$ is generated. For each $i \in \mathcal{I}$, the edge (i, x) in the matching is swapped with the edge $(P(i), y)$ in the matching if the resulting edges (i, y) and $(P(i), x)$ are still consistent. This gives a *seed matching*. Then the procedure applies another 10,000 iterations as before to generate the first sample. The iterations continue and for every 10,000 iterations, a new sample is generated. In this way, the procedure generates 250 samples. At this point, a new seed matching is re-generated *from scratch*, and another 250 samples are generated. For the

Dataset	# items	# Trans.	# Gps.	Size 1 Gps.
CONNECT	130	67557	125	122
PUMSB	2113	49046	650	421
ACCIDENTS	469	340184	310	286
RETAIL	16470	88163	582	218
MUSHROOM	120	8124	90	77
CHESS	75	3196	73	71

(a) General Statistics

Dataset	Mean	Median	Min.	Max.
CONNECT	0.0081	0.0029	0.000015	0.0519
PUMSB	0.00154	0.000041	0.00002	0.0536
ACCIDENTS	0.00324	0.000176	0.000029	0.04966
RETAIL	0.00099	0.0000113	0.0000113	0.30102
MUSHROOM	0.01124	0.00394	0.00049	0.1477
CHESS	0.01389	0.00657	0.000313	0.0494

(b) Frequency Gap Statistics

Fig. 10. General and Frequency Gap Statistics for Various Benchmarks

results reported below, we used 5,000 samples.

6.2 Evaluation for Items

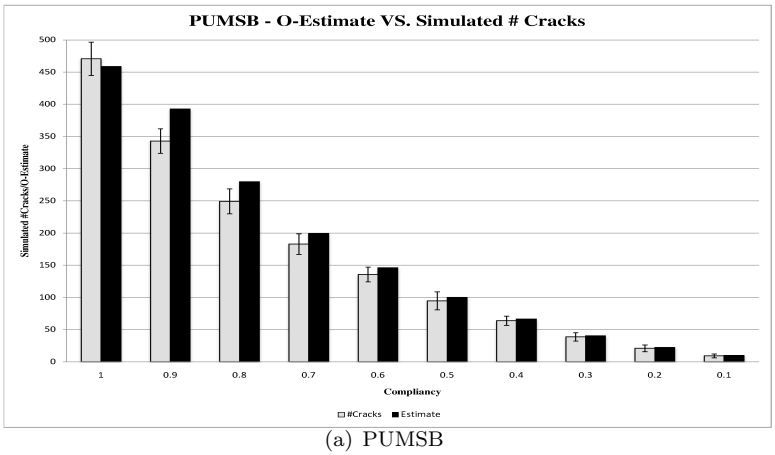
The remainder of this section is divided into two parts. The first part considers singleton itemsets, and the second part considers itemsets of arbitrary size.

6.2.1 *Accuracy of the O-Estimates.* Figures 11 and 12 compare the O-estimates with the estimates from simulation on the benchmark datasets for various compliancy values. Recall that the average simulated estimates is obtained from 5 runs. We also recorded the standard deviation on the simulated estimates. In almost all the cases, the differences between the O-estimates and the average simulated estimates are well within one standard deviation. This accuracy shown by the O-estimates also confirms that the recipe's choice of using δ_{med} as the width of the intervals works out well. Incidentally, even for the RETAIL dataset, it takes only a few seconds to compute the O-estimate.

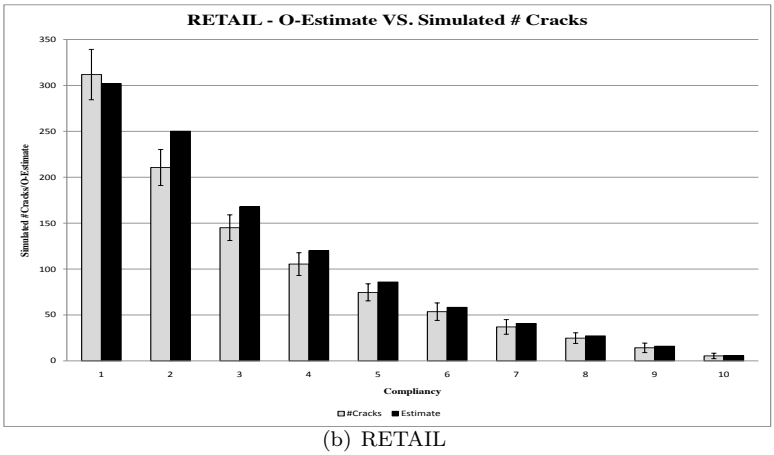
6.2.2 *The Effectiveness of the Recipe for Items.* Figure 13 illustrates how the proposed recipe in Figure 8 works for real datasets. The x-axis shows the values of α and the y-axis shows the O-estimates expressed in fractions of the domain size. Let us say that the data owner has a tolerance level of $\tau = 0.1$, as shown by the horizontal line. First, for the RETAIL dataset, it is a clear decision to release the anonymized data. In fact, even if the hacker correctly guesses all the frequency intervals, the expected fraction of cracks is still below 0.02.

The other datasets are different from RETAIL. For the PUMSB dataset, a tolerance level of 0.1 corresponds to a compliancy $\alpha_{max} \approx 0.7$. In other words, the hacker needs to have correctly guessed the intervals for 70% of items. This percentage is probably high enough to make the data owner feel secured in releasing the data. The situation is similar for the ACCIDENTS dataset. For CHESS and MUSHROOM, the compliancy returned is around 0.45 and 0.55 respectively and the decision to disclose is based on the comfort level of the data owner.

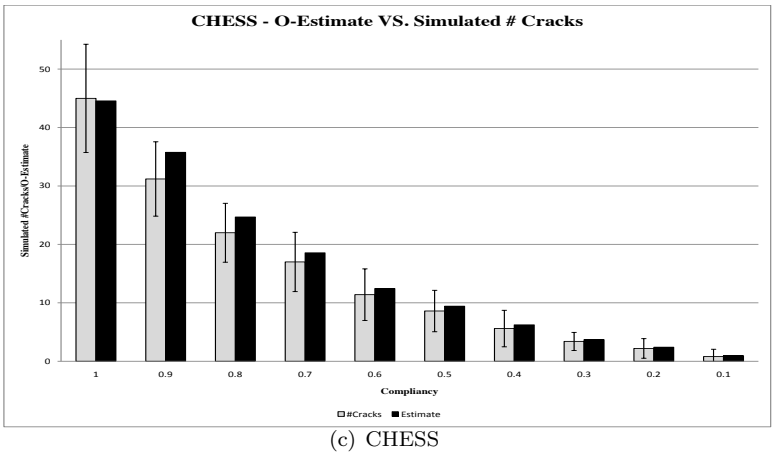
The situation for CONNECT is very different. The tolerance level of 0.1 corre-



(a) PUMSB

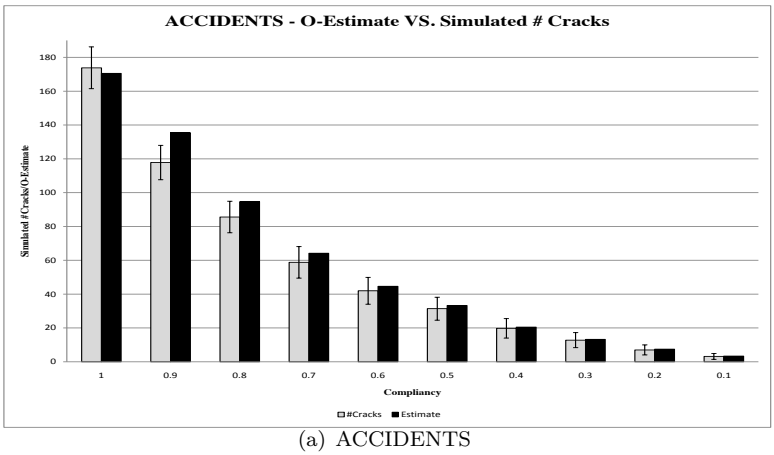


(b) RETAIL

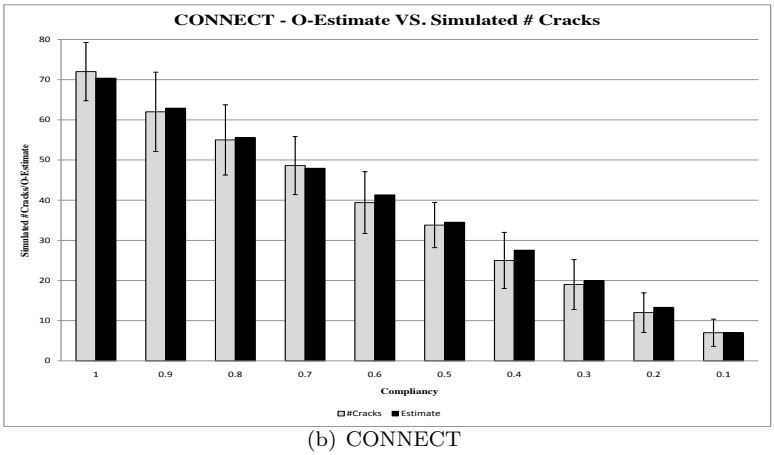


(c) CHESS

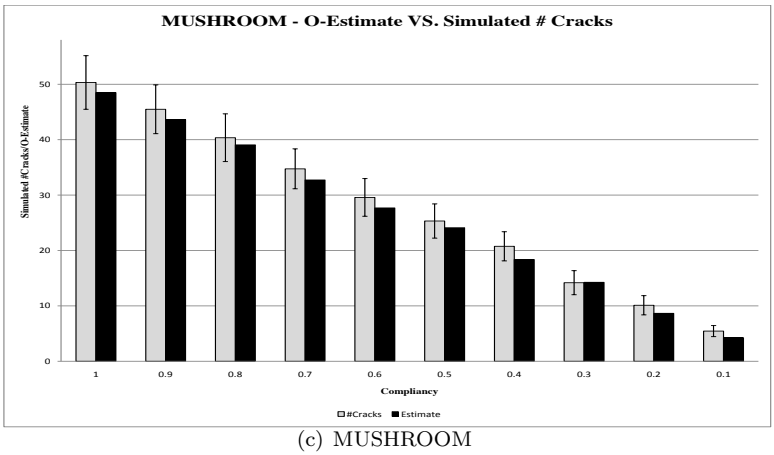
Fig. 11. O-estimates vs Average Simulated Estimates at various Compliancy levels for Benchmark Datasets



(a) ACCIDENTS



(b) CONNECT



(c) MUSHROOM

Fig. 12. O-estimates vs Average Simulated Estimates at various Compliancy levels for Benchmark Datasets

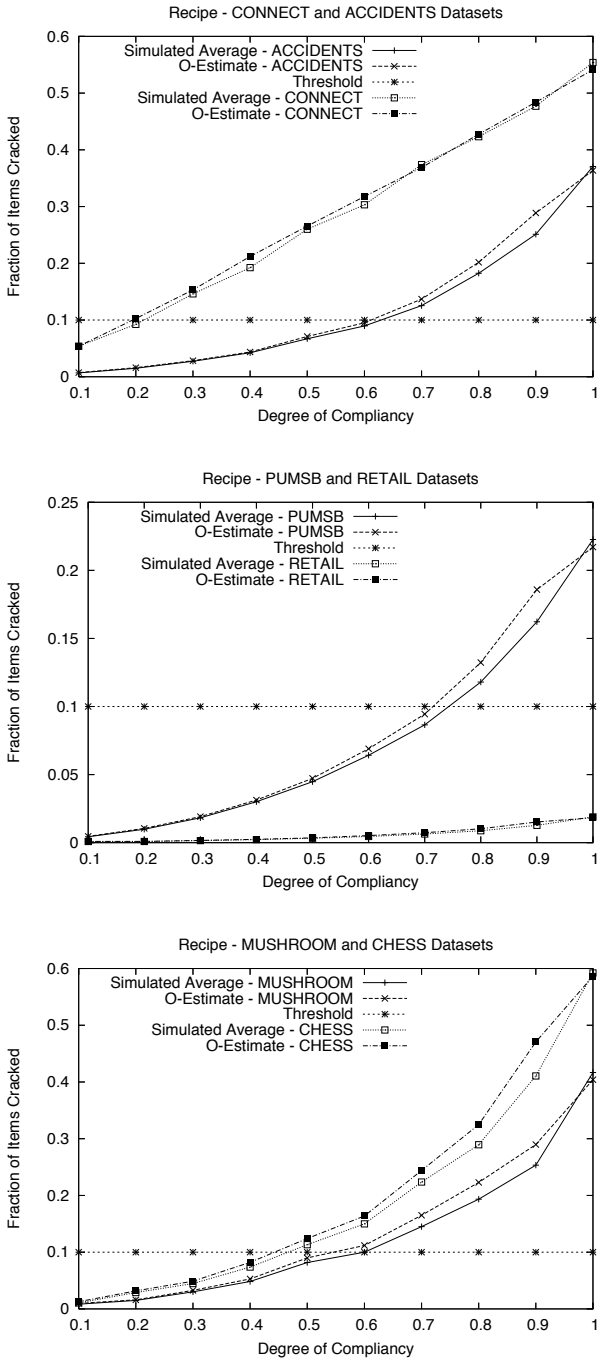


Fig. 13. Varying the Degree of Compliancy

sponds to $\alpha_{max} \approx 0.2$. Thus, the comfort level of the data owner is only about 20% of the items, which corresponds to 26 items to be exact. The data owner may want to think twice before releasing the data.

The shapes of the curves in Figure 13 are interesting. For RETAIL and CONNECT, the curves are approximately linear. Notice that CONNECT has a small number of items but a relatively large number of transactions, whereas the situation is rather contrary for RETAIL. Yet, their curves look similar. In contrast, the curves for PUMSB and ACCIDENTS are super-linear. Thus, how the O-estimates vary with the degree of compliancy does not appear to be determined directly by domain size or transaction size. Finally, Figure 13 also shows that the O-estimates and the simulated estimates are close to each other, confirming the accuracy of the O-estimates.

6.2.3 *Degrees of Compliancy from Similar Data.* As shown above, the data owner needs to wrestle with the decision whether α_{max} is high enough, particularly if the hacker may have gained partial information based on similar data. The question is how to define "similar" data. Below, we experiment with the idea that the data owner *simulates similarity by sampling*.

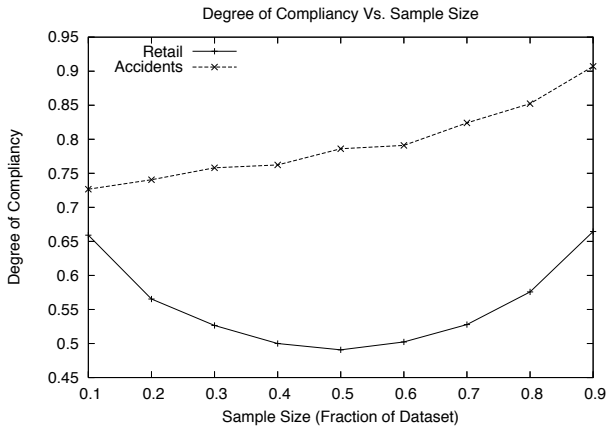


Fig. 14. Degrees of Compliancy from Similar Data

For the ACCIDENTS and RETAIL datasets, Figure 14 shows the variation in the degree of compliancy of a belief function created from samples of varying percentages. Rather surprisingly, the degrees of compliancy can be high even for small samples. For instance, for a sample of 10%, the α value is above 0.7 for ACCIDENTS. This information can be very significant for the data owner. For the ACCIDENTS dataset, recall from our earlier discussion based on Figure 13 that for a tolerance of $\tau = 0.1$, the corresponding $\alpha_{max} = 0.65$. Seeing that even a 10% sample, corresponding to a somewhat similar data set, can easily give an α value higher than 0.7, the owner may decide not to disclose the data after all.

In [Christopher Clifton 2000], Clifton argues that releasing a small random sample poses no threat to the data owner as little information can be revealed. In

the context of compliancy, this does not appear to be true for every dataset. In a follow up study, Ramesh [Ganesh Ramesh 2005] considers the effect of various parameters of sampling on the quality of knowledge that can be constructed from samples and concludes that even small samples released without replacement can yield prior knowledge of very high quality. The study considers various types of prior knowledge including belief functions and other types of parameters like *order* and *rank*.

As shown in Figure 14, for the RETAIL dataset, there is a gradual drop in compliancy as the sample size increases until the size reaches 50% of the original size. This is counter-intuitive on first sight. But there is an interesting subtlety here. For a normal dataset, like ACCIDENTS, as the sample size increases, the sampled median gap between frequency groups increases. Thus, the width of the intervals increases, making compliancy easier to satisfy. Hence, there is a gradual increase in compliancy as the sample size increases. In contrast, the RETAIL dataset is abnormally sparse. In particular, for a 10% sample, there are only about 8800 transactions over a domain of about 16,000 items. Thus, the items tend to cluster together as their frequencies are under-determined. As the sample size increases, some of the items in one frequency group, start to separate into more frequency groups. Thus, the sampled median gap between frequency groups drops, narrowing the intervals of the belief function. Hence, the compliancy drops accordingly. This phenomenon persists until the number of frequency groups stabilizes; from that point on, the normal trend kicks in. Incidentally, if instead of the sampled median, the sampled average gap is used as the width of the intervals in the belief function, the degree of compliancy is at 0.99 (not shown in the figure), uniformly across all sample sizes. This again confirms that using the average can be misleading.

We provide a simple procedure shown in Figure 15 that implements the idea of simulating similarity by sampling. It uses multiple samples of varying sizes to generate the kind of curves shown in Figure 14 for a given dataset. This curve can then be used in conjunction with the recipe in Figure 8. Specifically, the recipe returns the α_{max} value. The data owner can then use the curve to ascertain whether α_{max} is high enough based on the corresponding sample size value.

Procedure Similarity-by-Sampling
1. For a given range of sample sizes p
   a. Get a sample $\mathcal{D}_p$ of the database $\mathcal{D}$.
   b. Determine the frequency $\hat{f}_x$ $\forall x \in \mathcal{I}$ in $\mathcal{D}_p$.
   c. Determine the sampled median frequency gap δ'_{med} of the frequency groups in D_p.
   d. Determine the degree of compliancy α by checking if $\forall x \in \mathcal{I}$, $f_x \in [\hat{f}_x - \delta'_{med}, \hat{f}_x + \delta'_{med}]$.
   e. Repeat (a) to (d) for 10 samples and get the average α_p.

Fig. 15. Data Similarity by Sampling

6.3 Evaluation for General Itemsets

For a given interval belief function, the simulation generates many consistent matchings from the bipartite graph representing the crack space. These consistent matchings were generated exactly as described in Section 6.1. However, the only augmentation required was to compute the number of matchings in which the itemsets in

Θ were cracked. To do so, a vector of counts was initialized for the itemsets in Θ and for each consistent matching M generated by the simulation, the count for an itemset $X \in \Theta$ was incremented if X was cracked by M. This augmentation is shown in Figure 16. Once the simulation completes generating all the samples, the count for each itemset $X \in \Theta$ is divided by the number of samples. This gives the probability of the itemset X being cracked across all the samples. We call this probability the *simulated probability* of cracking itemset X. The OS estimate for the itemsets in Θ are computed using the procedure given in Figure 7.

For our experimental results, to keep the size of Θ manageable and to place the estimates on a fair ground for comparison, we consider the three examples Θ_0, Θ_1 and Θ_2 discussed in Section 2.3.

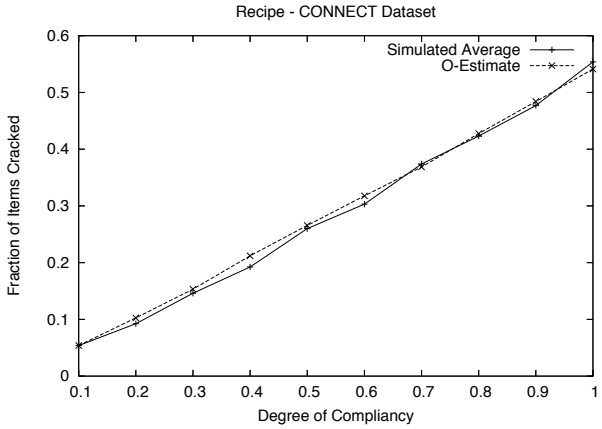
Procedure Count-Match
Inputs: M - A Consistent Matching; Θ - Itemsets of Interest, Count - A vector of size $|\Theta|$
1. For each $X \in \Theta$: If the itemset X is cracked in M
   Increment Count(X)

Fig. 16. Counting the Itemsets Cracked by a Matching

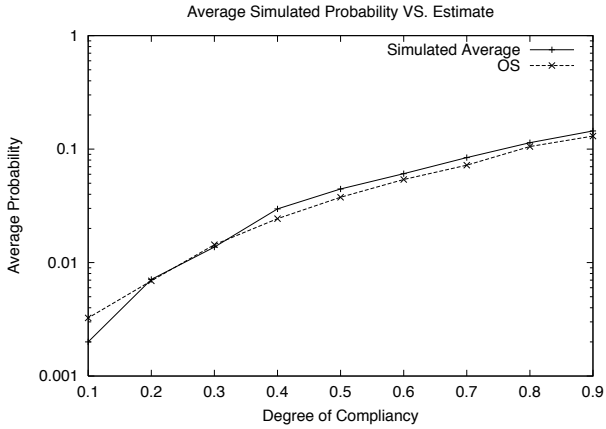
6.3.1 *Accuracy of the OS Estimate.* Figure 17 compares the OS estimate with the simulated probability (in log scale) on three benchmark datasets for various compliancy values of the belief functions. Note that the estimates and the simulated probability are averaged across all the pairs. The simulated probability and the estimates are themselves averaged based on values obtained from 5 runs. From the plots, it can be seen that for most of the compliancy values the average OS heuristic value is a fairly accurate estimate of the average simulated probability. For the lowest compliancy value of 0.1, the OS heuristic overestimates the simulated probability. However, for the most part, the margin of error in estimation for the OS estimate was around 10%.

The tables in figure 18 show the comparison of the simulated probability with the OS estimate for k-itemsets for $k > 3$, for two of the benchmark datasets (ACCIDENTS and MUSHROOM). We used fully compliant interval valued belief functions for this experiment. To keep the itemsets of a manageable size, we chose 10000 random k-itemsets for $k = 4, 5, 6$ respectively, and measured the average simulated probability and OS estimate. These average values are computed for a run of 500 samples each and are further averaged across 5 runs. From the values in the tables, we observe that the OS estimate is almost always larger than the simulated probability, sometimes by a factor of 10. Recall from Figure 17, that the OS estimate for 2-itemsets is fairly accurate. From the table, we observe a general trend that the OS estimate deteriorates as the itemset size increases. However, it is worth noting that the OS estimate is often an overestimate of the simulated probabilities, resulting in a conservative decision for the data owner.

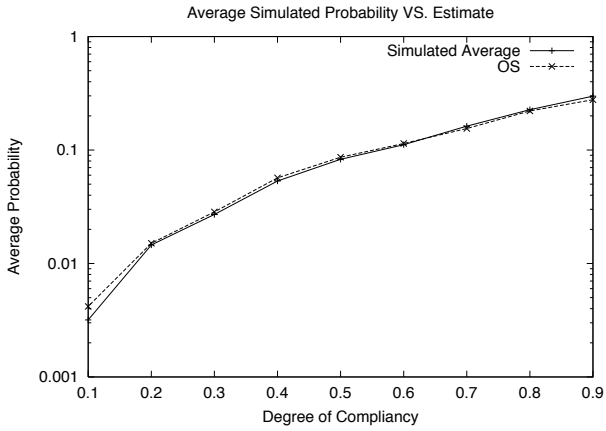
6.3.2 *Evaluating The Recipe for Itemsets.* Figures 19,20 and 21 demonstrate the effectiveness of the recipe for three benchmark datasets. All the plots have the compliancy values in the x-axis and the fraction of itemsets in Θ that are cracked in the y-axis. We consider the three examples discussed in Section 2.3 as the set of itemsets of interest. The first collection, Θ_0, is the set of all pairs of items.



(a) CONNECT dataset



(b) MUSHROOM dataset



(c) CHESS dataset

Fig. 17. Simulated Probability VS. *OS* estimate averaged over all 2-itemsets

K	Simulated	*OS* Estimate
4	0.005346	0.016249
5	0.001008	0.005076
6	0.0003626	0.00193

K	Simulated	*OS* Estimate
4	0.00196	0.0304
5	0.000335	0.0121
6	0.00029	0.0057

Fig. 18. Average OS Estimate Comparison for k-Itemset Collections ($k > 3$) for ACCIDENTS and MUSHROOM

The second collection Θ_1, is the set of all pairs of items where the items are not among the top 10% of the most frequent items. The third collection Θ_2 is the same as Θ_1 except the items are not among the top 20% of the most frequent items. We experimented with various values of thresholds τ and σ but for the sake of illustration, we present results for $\sigma = 0.5$ and $\tau = 0.1$. That is, the data owner has a privacy breach tolerance of 10%, where he regards as a breach a probability of crack of 50% or more for those items (s)he is interested in.

Let us first consider the recipe on the CONNECT dataset shown in Figure 19. For Θ_0 and Θ_1, the compliancy value returned when using the *OS* estimate is around 0.6 while using the actual simulated probability value would have returned a higher value (0.65 or closer to 0.7), as seen from the intersection of the threshold line (at 0.1) with these curves. For these two cases, the *OS* estimate is more *conservative* than the actual simulated probability value, as the data owner gets the picture that a hacker's belief function needs to be compliant on only 60% of the items (as opposed to a higher value specified by the simulated average) to breach the tolerance requirement. The same conclusions, more or less, can be drawn in analyzing Figure 20 for the CHESS dataset.

It is important to note that the purpose of the recipe is to determine a yes or no answer to releasing data. As observed in all three plots in Figure 19, even though the *OS* estimate is not perfect, the data owner makes the right yes decision when the compliancy is less than 0.6 and the right no decision when the compliancy is greater than 0.7. Obviously, these values depend on the tolerance threshold chosen by the data owner. However, the general point is as follows: *Even though, the OS estimate may not be accurate over a wide range of α-compliancy, the subrange within which the data owner makes the wrong decision is significantly smaller.*

For the MUSHROOM dataset in Figure 21, it is interesting to study why the *OS* estimate turns from a *conservative* value in Figure 21(a) to a very liberal value in Figures 21(b) and 21(c), for high values of compliancy. Note that for Θ_0, we consider all pairs of items. When we consider the set of itemsets in Θ_1, the pairs involving the top 10% of the frequent itemsets are omitted from consideration and those involving the top 20% of items are omitted when considering Θ_2. If the *OS* estimate is accurate for the pairs that are very frequent, then there is a considerable loss in accuracy when these pairs are suddenly omitted from consideration. This can happen when there are a large number of singleton groups in the higher end of the frequency spectrum and these groups are well separated. This accounts for the jump in the *OS* estimate. For the lower compliancy values, the *OS* estimate is fairly consistent in being conservative. However, once again, for the tolerance threshold of 0.1, the underestimation does not pose any significant impact on the owner's decision to release the dataset. It is only when the data owner lowers the threshold to a very low level such as 0.02, we see a wider subrange of compliancy

values leading to a wrong decision.

7. HANDLING KNOWLEDGE OF CORRELATION BETWEEN ITEMS

Thus far, our risk analysis is modeled based on the prior knowledge a hacker has on the frequency of individual items in the domain. In practice, the hacker may have other types of knowledge about the domain. In this section, we consider prior knowledge in which the hacker (in addition to knowledge of single items and their frequency) also has some knowledge of which items occur together in the dataset. For example, the hacker may know that hamburger and ketchup occur together 90% of the time, while hammer and lipstick rarely occur together. We refer to such information as *correlation knowledge*.[5] In particular, the first example is an instance of positive correlation while the second is an instance of negative correlation. We use positive correlation to highlight our approach for integrating this prior knowledge into our existing framework of disclosure risk analysis. Specifically, we first discuss possible attack models by a hacker based on correlation knowledge. We then explain how the proposed framework can be easily adapted to apply the heuristic estimates. At the end of this section, we briefly discuss how negative correlation can be handled as well.

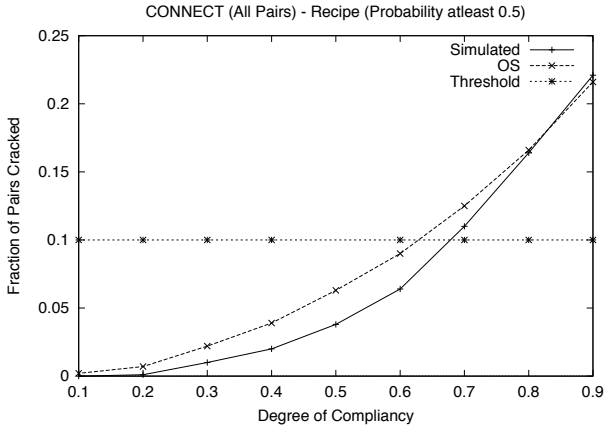
7.1 Using Positive Correlation to Refine the Bipartite Graph

Let us consider the simplest case. Suppose that a hacker knows that items 1 and 2 occur together in every transaction and from the anonymized dataset, the hacker computes the frequency of pairs of items and discovers that items $1'$ and $2'$ are the only ones that occur together all the time. The hacker then associates the pair of items $(1, 2)$ with the pair of anonymized items $(1', 2')$. In this case, not only is the itemset $\{1, 2\}$ cracked but this knowledge can also be used as follows. In the space of crack mappings, the items 1 and 2 map only to the anonymized items $1'$ and $2'$. All other edges other than those between $(1, 2)$ and $(1', 2')$ can be removed from the space of crack mappings. Consider the space of crack mappings in Figure 3(a) that correspond to the belief functions f and h in Figure 2. In this context, the knowledge that $(1, 2)$ identifies with the pair $(1', 2')$ will lead to the removal of the edges $(1', 3)$, $(1', 4)$, $(1', 6)$, $(3', 1)$, $(4', 1)$ and $(6', 1)$ in the bipartite graph corresponding to the belief function f and the removal of the edges $(1', 3)$, $(1', 4)$, $(1', 6)$, $(2', 4)$, $(2', 5)$, $(3', 1)$, $(3', 2)$, $(4', 1)$, $(4', 2)$, $(5', 1)$, $(6', 1)$ and $(6', 2)$ in the bipartite graph corresponding to the belief function h. Additional items may get cracked as a result of the removing these edges.

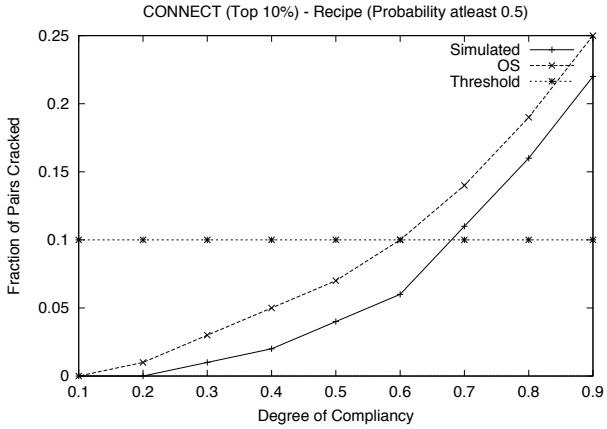
The above example is an extreme case where the hacker is able to exactly associate a pair of correlated items with a pair of correlated anonymized items. In practice, there may be many pairs of anonymized items with a high frequency of co-occurrence [6], leading to the hacker associating the pair $(1, 2)$, e.g., with, say k pairs of anonymized items (i'_{11}, i'_{12}), (i'_{21}, i'_{22}),..., (i'_{k1}, i'_{k2}). Then, all edges be-

[5]The reader should note that, by correlation, we mean the *co-occurrence* of items in a transaction. The conventional statistical definition of correlation coefficient has a different meaning.

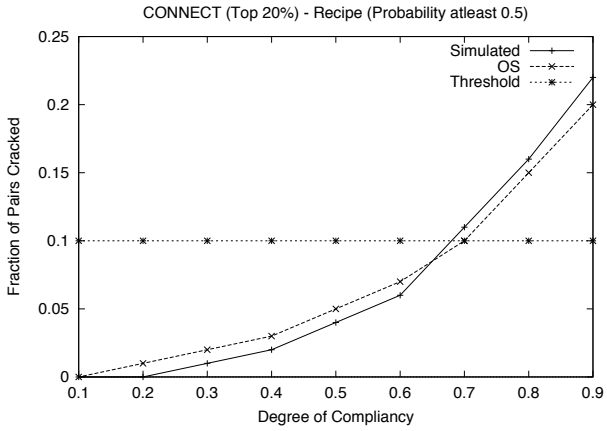
[6]In the context of frequent itemsets, a higher order correlation can be handled with pairs of correlated items. For example, if we know that (1,2,3) are correlated, this can be modeled with the pairs (1,2), (2,3) and (1,3).



(a) Θ_0 - All Pairs, $\sigma = 0.5$, $\tau = 0.1$

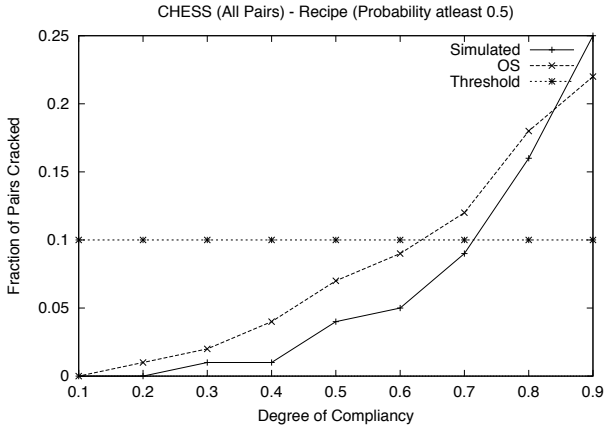


(b) Θ_1, $\sigma = 0.5$, $\tau = 0.1$

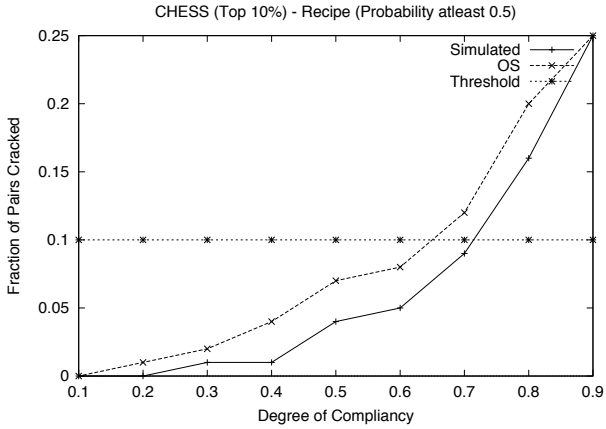


(c) Θ_2, $\sigma = 0.5$, $\tau = 0.1$

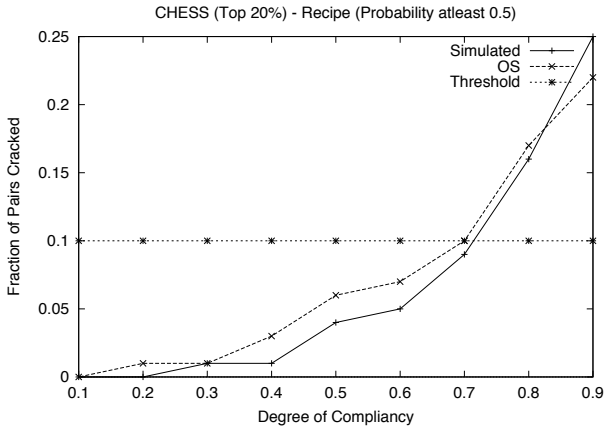
Fig. 19. Recipe for the CONNECT dataset



(a) Θ_0 - All Pairs,$\sigma = 0.5$,$\tau = 0.1$

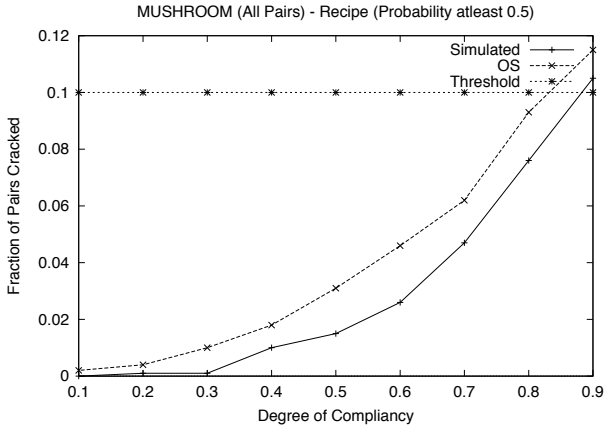


(b) Θ_1, $\sigma = 0.5$, $\tau = 0.1$

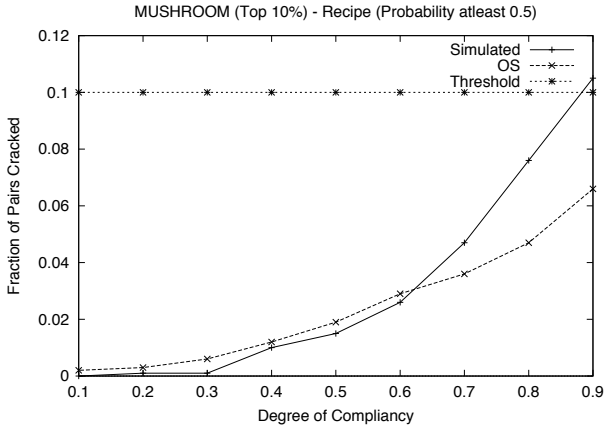


(c) Θ_2, $\sigma = 0.5$, $\tau = 0.1$

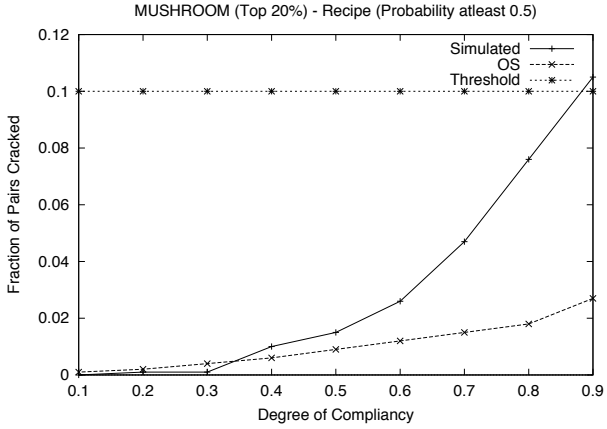
Fig. 20. Recipe for the CHESS dataset



(a) Θ_0 - All Pairs, $\sigma = 0.5$, $\tau = 0.1$



(b) Θ_1, $\sigma = 0.5$, $\tau = 0.1$



(c) Θ_2, $\sigma = 0.5$, $\tau = 0.1$

Fig. 21. Recipe for the MUSHROOM dataset

tween the items $(1, 2)$ and any anonymized item that is not i'_{ij}, for $i = 1, 2, \ldots, k$ and $j = 1, 2$, can be removed from the bipartite graph of crack mappings. This process is formally illustrated in Figure 22 as *Procedure FilterSinglePairEdges*.

Procedure FilterSinglePairEdges
Inputs: Pair (i, j), $i, j \in \mathcal{I}$, Crack Bipartite Graph $G = (\mathcal{I} \bigcup \mathcal{J}, E)$, Anonymized item Pairs (a'_u, b'_u) for $u = 1, 2, \ldots, k$
Output: New Crack Bipartite Graph G after filtering edges
1. Let $A' = \{a'_1, b'_1, \ldots, a'_k, b'_k\}$.
2. For each $x' \in \mathcal{J} - A'$
   a. If $(i, x') \in E$, $E = E - \{(i, x')\}$.
   b. If $(j, x') \in E$, $E = E - \{(j, x')\}$.

Fig. 22. Removing edges using correlation knowledge of a single pair of items

A hacker may also have very general knowledge about co-occurrence of pairs of items. For example, the hacker may know that the pairs $(1, 2)$, $(13, 15)$ and $(20, 25)$ always occur together in *almost all* the transactions but may not have an exact value for these co-occurrences. In such cases, it is reasonable for the hacker to map these item pairs to the high frequency pairs of anonymized items computed from the database[7]. In this case, the hacker may repeatedly apply *procedure FilterSingle-PairEdges* on each pair of items with the set of high frequency pairs of anonymized items identified from the database.

Procedure IdentifyAndFilterEdges
Inputs: Pairs (a_i, b_i), $a_i, b_i \in \mathcal{I}$, for $i = 1, 2, \ldots, w$ that are believed to co-occur frequently, Crack Bipartite Graph $G = (\mathcal{I} \bigcup \mathcal{J}, E)$, Anonymized Dataset D'
Output: New Crack Bipartite Graph G after filtering edges
1. Compute the frequency/correlation between all pairs of anonymized items from D'.
2. Identify the top k most frequent pairs of anonymized items as candidates for the w input pairs (a_i, b_i). Call this set A.
3. For each (a_i, b_i), $i = 1, 2, \ldots, w$
   a. Call FilterSinglePairEdges using (a_i, b_i), A and G.

Fig. 23. Identifying and Matching frequent co-occurring pairs and Removing Edges

Procedure IdentifyAndFilterEdges in Figure 23 gives a procedure using which a hacker can identify k candidate pairs of anonymized items and match them with the w pairs of items which are believed to co-occur frequently. Using these candidates, the hacker then repeatedly applies the *procedure FilterSinglePairEdges* to remove edges in the crack bipartite graph. Note that k can be greater than or equal to w.

We have looked at one kind of usage of knowledge of co-occurrence. The hacker may also use knowledge of co-occurrence to refine the belief of frequency of items. To illustrate this, let us consider an example. Suppose that the hacker knows that whenever item 1 occurs in a transaction, item 2 almost always occurs and vice versa. This knowledge will lead to the conclusion that the frequency of items 1 and

[7] A threshold may be used to identify which set of pairs of anonymized items are candidates.

Procedure RefineBeliefs
Inputs: Pair of items (a, b) that co-occur frequently, β - Belief Function
Output: New Belief Function with refined belief for a and b
1. If $\beta(a) \bigcap \beta(b) \neq \emptyset$
   a. $\beta(a) = \beta(a) \bigcap \beta(b)$.
   b. $\beta(b) = \beta(a)$.

Fig. 24. Refining Belief Frequencies of co-occurring item pairs

2 in the database are roughly the same. Let us assume that the belief intervals for 1 and 2 are $[l_1, r_1]$ and $[l_2, r_2]$ respectively. The hacker can refine the belief about the frequency of items 1 and 2 to the intersection of their belief intervals[8]. This is illustrated in Figure 24 in *procedure RefineBeliefs*. This revision may in turn translate into removal of more edges from the crack bipartite graph.

Once the crack bipartite graph is pruned (or the prior beliefs are refined), all the previously proposed heuristic estimation techniques can be applied without any change.

7.2 Changing the Recipe

There remains one last piece in the analysis. How does the *data owner* model the correlation knowledge of the hacker? From a data owner's point of view, the correlation knowledge possessed by the hacker is unknown. We could take a stance that is similar to the single item case and start with a paranoid model where the data owner assumes that the hacker knows the co-occurrence value of every single pair of items and can correctly map it to the anonymized item pairs. However, this is unrealistic[9]. For the remainder of this section, we consider the situation when the data owner assumes that the hacker's knowledge of the w pairs of correlated items are exactly the top k most co-occurring pairs. As it is, the hacker is already very knowledgeable. We do not consider an even stronger case when the hacker knows the exact rank of the top-k most frequently co-occurring pairs of items.

Figure 25 gives a modified recipe for risk assessment. The first 9 steps are similar to the corresponding steps in Figure 8. These steps use the belief function on frequency of items to determine the risk of disclosing the data. When this risk is tolerable, then the recipe checks to see whether additional items can be breached due to correlation knowledge starting from step 11. As discussed in the previous subsection on refining the crack bipartite graph, this estimation requires a value for the parameter k. This value could be input by the data owner who may decide to try different values of k before making a decision.

[8]If the intersection is empty, then the prior knowledge may be inconsistent and there are multiple ways in which the refinement can be done. For the purposes of this paper, we will assume that the refinement is done only when the intersection is nonempty and the belief functions are not altered when the intersection is empty.

[9]Even for the single item case it is unrealistic to expect the hacker to know exact frequencies of all the items. The number of pairs is much larger than the number of single items (order of square of the number of items) and it is very unlikely that a hacker will exactly know this information for each pair.

Algorithm Assess-Risk-Corr
Inputs: τ - degree of tolerance; $\mathcal{D}$ - Anonymized Database; k - Positive Integer
1. Compute g based on Lemma 3.5.
2. If $g \leq (\tau \times |\mathcal{I}|)$, go to step 11.
3. Compute the frequency groups from $\mathcal{D}$, and the median gap M between frequency groups
4. Set width δ_{med} to be M.
5. Set up $\beta(x) = [f_x - \delta_{med}, f_x + \delta_{med}]$ for $x \in \mathcal{I}$, where f_x denotes the frequency of $x \in \mathcal{I}$.
6. Compute the O-estimate $OE(\beta, \mathcal{D})$ according to Figure 4.
7. If $OE(\beta, \mathcal{D}) \leq (\tau \times |\mathcal{I}|)$, go to step 11.
8. Set α to be 1.
9. Perform a binary search on α to determine the largest α so that the corresponding α-compliant belief function β satisfies the condition $OE(\beta, \mathcal{D}) \leq (\tau \times |\mathcal{I}|)$.
10. Return the value of α. If α is found to be reasonable, go to step 11.
11. Compute the top k most frequently occurring pairs in $\mathcal{D}$.
12. Use these pairs to filter edges from the bipartite graph of cracks mappings (or to refine the belief functions).
13. Compute the new O-estimate. If this is $\leq (\tau \times |\mathcal{I}|)$ then disclose D else do not disclose D.

Fig. 25. The Modified Recipe for Risk Assessment

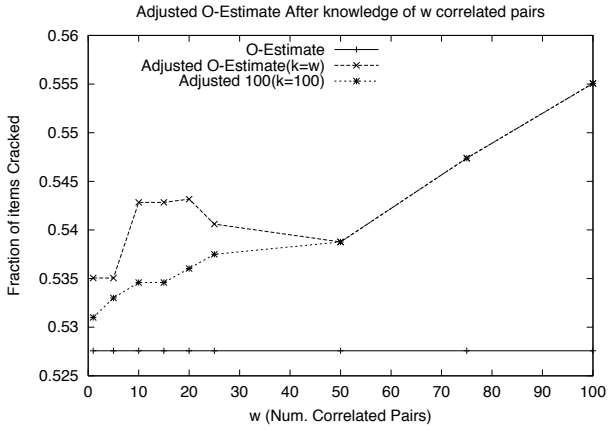
7.3 Experimental Evaluation of the Impact of Correlation Knowledge of Pairs

Below, we present experimental results on how knowledge of correlation affects the O-estimate values. Figure 26 shows two plots for the MUSHROOM and CONNECT datasets. Each plot shows the original O-estimate and the adjusted O-estimate when the hacker has knowledge of *all* the top $w(= k)$ most frequently co-occurring pairs in the dataset. This is labeled as "Adjusted O-estimate" in the plots. The O-estimate labeled "Adjusted 100" (and referred to as adjusted 100 O-estimate in our discussion) is computed by removing the edges when the w pairs of known correlated pairs of items are matched with the top 100 correlated pairs from the anonymized data.

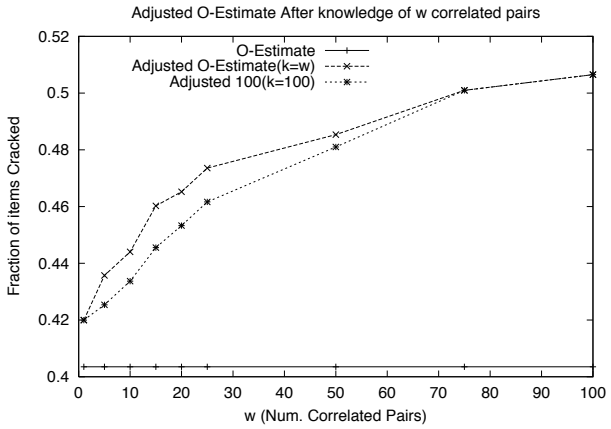
One would expect the adjusted O-estimate and adjusted 100 O-estimate to increase as the value of w increases. While this general trend is true, there are two interesting observations. First, sometimes the adjusted O-estimate remains the same as w increases. For example, for the MUSHROOM dataset, the top 8 most frequently occurring pairs of items are (in order): $(85, 86)$, $(34, 85)$, $(34, 86)$, $(85, 90)$, $(34, 90)$, $(86, 90)$, $(36, 85)$, $(36, 86)$. When $w = 1 = k$, the hacker knows the pair of items $(85, 86)$ and when $w = 2 = k$, the hacker knows the pairs $(85, 86)$ and $(34, 85)$. Since there is one new item 34 whose outgoing edges may be removed, the O-estimate may increase from $w = 1$ to $w = 2$. Consider what happens when $w = 3 = k$. The addition of the knowledge of the new itemset $(34, 86)$ does not cause any new edges to be deleted. Hence, the adjusted O-estimate does not increase when w changes from 2 to 3.

The second observation is that when w increases from 20 to 25 for the CONNECT dataset, the adjusted O-estimate decreases. This decrease can be attributed to the following reason. When w is 20, the matching of the original item pairs was with the top 20 pairs of anonymized items. Hence, the edges between these items and the anonymized items occurring in the pairs from 21 through 25 will be removed. However, when w is 25, the edges between these pairs will be retained and hence the O-estimate might actually decrease, as seen in this instance. Thus, in general, the extent of disclosure risk does not necessarily have a monotonic behavior as w increases. However, as shown by the curves for the adjusted 100 estimate, the monotonic behavior is preserved i.e., when $k > w$ and is kept constant, increasing w leads to increasing disclosure risk.

Most importantly, it is interesting to note that even when the hacker knows all the top 50 correlated pairs of items, the additional disclosure caused by this knowledge is increased by only a small percentage - e.g. 2% for the CONNECT dataset (increase of 1.5 items out of 130) and 7% for the MUSHROOM dataset (increase of 8 items out of 120). We believe that for a universe of a little more than 100 items, knowing the top 50 correlated pairs is perhaps unrealistically conservative. Contrary to our belief, this observation shows that knowing correlated knowledge may not have too significant an impact on disclosure.



(a) CONNECT dataset



(b) MUSHROOM dataset

Fig. 26. Original VS. Adjusted O-Estimates for Single Items With Knowledge of Top w frequently Co-occurring item pairs

So far, we have mainly focused on *positively* correlated pairs. What can we say about negative correlation? Let us take an example – suppose the hacker knows hammer (item #1) and lipstick (item #2) rarely occur. Suppose we want to combine this knowledge of negative correlation with frequently co-occurring pairs of

anonymized items, to possibly prune edges from the crack bipartite graph. Let us suppose that $(3', 4')$ is a frequently co-occurring pair of anonymized items. Can we eliminate edges matching $(1, 2)$ to the set of items $\{3', 4'\}$ based on this knowledge? It turns out that doing so is not sound. All we know is that not both of the edges representing $3' \mapsto 1$, $4' \mapsto 2$ can occur simultaneously. The problem is either one of them can occur separately and we do not know which one. The general point is that dealing with negative correlation requires reasoning at the level of mappings rather than at the level of edges. We explore this in future work.

8. DISCUSSION

8.1 Beyond Frequent Sets

Much of the results presented here can carry over to situations other than frequent itemset mining, as long as the bipartite graph representing the space of crack mappings is set up by some means. Specifically, Lemmas 3.2 to 3.6 are already expressed in terms of the underlying bipartite graph. At present, Lemmas 4.2 and 4.4 are expressed in terms of the belief functions, mainly for ease of understanding. However, they can re-stated solely from the perspective of the underlying bipartite graph. As an example, Lemma 4.2 can be restated as follows:

Let G_1 and G_2 be bipartite graphs with the vertex set $\mathcal{I} \cup \mathcal{J}$ and edge sets E_1 and E_2 respectively, modeling the space of crack functions β_1 and β_2. Then, if G_1 is a subgraph of G_2, then the O-estimate for G_2 is smaller than that of G_1.

As an example of the application of our framework to the analysis of other patterns, consider the data disclosure dilemma in the context of classification. The data owner needs to decide whether to release an anonymized relation with attributes: age, ethnicity and car-model. Let us say that the real domain consists of people identified by their names (e.g. {Bob,Mary,... }), and the anonymized domain identified by an integer {1', 2', ... }. Suppose that the hacker has partial information about certain individuals. Recall from Figure 14 that even a small sample of 10% can reveal a lot of true information. In any case, regardless of how this piece of partial information comes about, if the hacker somehow knows that John is a Chinese owning a Toyota, then edges can be set up between (x', John) for all anonymized items x' with ethnicity being Chinese and car-model being Toyota. Similarly, if the hacker somehow knows that Mary's age is between 30 and 35, the appropriate edges can be set up in the bipartite graph to connect Mary to all anonymized items x' in the same age group. And if the hacker has no knowledge of Bob, Bob is connected to every anonymized item in the graph. Once the graph is set up, we can re-apply the framework and the recipes presented here to help the data owner to decide whether it is safe to release the anonymized relation.

8.2 Summary and Ongoing Work

A classic dilemma that an organization faces is if the data is not released, they cannot take advantage of the opportunities offered by data mining. On the other hand, if they do, they run the risk of disclosing sensitive data. To mitigate the latter, they may sanitize their data or more generally apply some data transformations. In this paper, we address the question of how safe the anonymized data is with respect to protecting the true identities of the data items. The novelty of our work

is to incorporate the possible existence of partial background knowledge possessed by the hacker, an issue mostly overlooked in the literature. We propose various classes of belief functions, representing the capturing of various degrees of partial information. We address the safety question both for single items and for sets of items. We derive exact or approximate formulas for computing the expected number of cracks and similarly for computing the probability of a given itemset being cracked. In particular, we propose the O-estimate heuristic for single items and the OS-estimate for itemsets which are applicable to any bipartite graph/belief function, and are easy to compute. We evaluate the accuracy of O-estimates by detailed experimentation with real datasets. Our results show that our framework is effective and that while exact analysis may be very hard, disclosure risk analysis based on heuristics is promising. Last but not the least, we also show how prior knowledge of positive correlation between pairs of items can be incorporated into our framework of disclosure risk analysis.

As observed from the experimental results, disclosure risk is closely connected with the characteristics of the dataset and the quality of prior knowledge. The data owner needs to balance the benefits of releasing data with potential disclosure risk. We consider the recipe developed in this paper as just one out of a suite of tools that we consider essential to assist the data owner in making his decision. In particular, the data may be released for mining different kinds of patterns. Thus tools that enable disclosure risk analysis w.r.t. different patterns are crucial. Furthermore, data owner may perform transformations other than anonymization before releasing the data and thus the "tool box" needs to be able to handle a rich set of transformations in conducting the analysis. Finally, in a multi-party consortium, an interesting topic is the development of protocols to conduct anonymization. In some applications, local anonymization would be sufficient. However, in other applications, a more sophisticated protocol may be necessary. These are interesting issues to explore in future.

REFERENCES

A. Meyerson and R. Williams. 2004. On the complexity of optimal k-anonymity. In *ACM PODS Conference*. 223–228.

Alan G. Konheim. 1981. *Cryptography: A Primer*. John Wiley and Sons, New York.

Alastair R. Beresford and Frank Stajano. 2004. Mix zones: User privacy in location-aware services. In *Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMM)*. 127–131.

Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. 2004. Privacy preserving mining of association rules. *Information Systems 29,* 4, 343–364.

Charu C. Aggarwal and Philip S. Yu. 2004. A condensation approach to privacy preserving data mining. In *EDBT*. 183–199.

Christopher Clifton. 2000. Using sample size to limit exposure to data mining. *Journal of Computer Security 8,* 4, 281–307.

Daksha Agrawal and Charu C. Aggarwal. 2001. On the design and quantification of privacy preserving data mining algorithms. In *ACM PODS Conference*. 247–255.

FIMI. 2003. The fimi repository at [http://fimi.cs.helsinki.fi/fimi03/].

Gagan Aggarwal, Tomas Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. 2005. Anonymizing tables. In *ICDT Conference*. 246–258.

Ganesh Ramesh. 2005. Can attackers learn from samples? In *VLDB Workshop on Secure Data Management (SDM)*. 104–123.

Gehrke, J. 2005. Models and methods for privacy-preserving data publishing and analysis: invited tutorial. In *PODS*. 316.

Joseph Domingo-Ferrer, Anna Oganian, and Vincent Torra. 2002. Information-theoretic disclosure risk measures in statistical disclosure control of tabular data. In *IEEE SSDBM Conference*. 227–231.

Joseph Domingo-Ferrer and Vincent Torra. 2002. Validating distance-based record linkage with probabilistic record linkage. In *Catalonian Conference on AI (CCIA)*.

K. Muralidhar and R. Sarathy. 1999. Security of random data perturbation methods. *ACM TODS 24,* 4, 487–493.

L. Sweeney. 2002. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10,* 5, 557–570.

Laks V.S. Lakshmanan, Raymond Ng, and Ganesh Ramesh. 2005. To do or not to do: The dilemma of disclosing anonymized data. In *ACM SIGMOD Conference*. 61–72.

L.E. Rasmussen. 1994. Approximating the permanent: A simple approach. *Random Structures and Algorithms 5,* 2, 349–361.

Leslie G. Valiant. 1979. The complexity of computing the permanent. *Theoretical Computer Science 8*, 189–201.

Mark Jerrum, Alistair Sinclair, and Eric Vigoda. 2001. A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. In *ACM STOC Conference*. 712–721.

Mark Jerrum and Umesh Vazirani. 1996. A mildly exponential approximation algorithm for the permanent. *Algorithmica 16,* 4-5, 392–401.

Murat Kantarcioglu and Christopher Clifton. 2004. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE TKDE 16,* 9, 1026–1037.

N.R. Adam and J.C. Wortmann. 1989. Security-control methods for statistical databases: A comparative study. *ACM Computing Surveys 21,* 4, 515–556.

P. Samarati and L. Sweeney. 1998. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. In *IEEE Symposium on Research in Security and Privacy*.

Rakesh Agrawal and Ramakrishnan Srikant. 2000. Privacy preserving data mining. In *ACM SIGMOD Conference*. 439–450.

Rakesh Agrawal, Tomasz Imielinksi, and Arun Swami. 1993. Mining association rules between sets of items in large databases. In *ACM SIGMOD Conference*. 207–216.

Raymond T. Ng, Laks V.S. Lakshmanan, Jiawei Han, and Alex Pang. 1998. Exploratory mining and pruning optimizations of constrained association rules. In *ACM SIGMOD Conference*. 13–24.

Richard A. Moore. 1996. Controlled data-swapping techniques for masking public use microdata sets. In *Statistical Research Division Report Series, RR 96-04, US Bureau of Census, Washington D.C.*

S. Hettich and S.D. Bay. 1999. The uci kdd archive [http://kdd.ics.uci.edu]. University of California, Irvine, CA.

Stephen E. Fienberg and A.B. Slavkovic. 2005. Preserving the confidentiality of categorical statistical databases when releasing information for association rules. *Data Mining and Knowledge Discovery Journal 11,* 2, 155–180.

Stephen E. Fienberg, Udi E. Makov, and Russell J. Steele. 1998. Disclosure limitation using perturbation and related methods for categorical data. *Journal of Office Statistics 14*, 385–397.

Vassilios S. Verykios, Ahmed K. Elmagarmid, Elisa Bertino, Yucel Saygin, and Elena Dasseni. 2004. Association rule hiding. *IEEE TKDE 16,* 4, 434–447.

Vijay S. Iyengar. 2002. Transforming data to satisfy privacy constraints. In *ACM KDD Conference*. 279–288.

William E. Winkler. 1993. Matching and record linkage. In *Research Report, U.S. Census Bureau*.

William E. Winkler. 1994. Advanced methods for record linkage. In *Research Report, U.S. Census Bureau*.

William E. Winkler. 1999. The state of record linkage and current research problems. In *Research Report, U.S. Census Bureau*.

X. Yang and Chen Li. 2004. Secure xml publishing without information leakage in the presence of data inference. In *VLDB Conference*. 96–107.