Relationship Privacy: Output Perturbation for Queries with Joins*

Vibhor Rastogi †
CSE, University of Washington
Seattle, WA, USA, 98105-2350
vibhor@cs.washington.edu

Michael Hay
CS, UMass Amherst
Amherst, MA, USA, 01003-9264
mhay@cs.umass.edu

Gerome Miklau
CS, UMass Amherst
Amherst, MA, USA, 01003-9264
miklau@cs.umass.edu

Dan Suciu
CSE, University of Washington
Seattle, WA, USA, 98105-2350
suciu@cs.washington.edu

ABSTRACT

We study privacy-preserving query answering over data containing relationships. A social network is a prime example of such data, where the nodes represent individuals and edges represent relationships. Nearly all interesting queries over social networks involve joins, and for such queries, existing output perturbation algorithms severely distort query answers. We propose an algorithm that significantly improves utility over competing techniques, typically reducing the error bound from polynomial in the number of nodes to polylogarithmic. The algorithm is, to the best of our knowledge, the first to answer such queries with acceptable accuracy, even for worst-case inputs.

The improved utility is achieved by relaxing the privacy condition. Instead of ensuring strict differential privacy, we guarantee a weaker (but still quite practical) condition based on adversarial privacy. To explain precisely the nature of our relaxation in privacy, we provide a new result that characterizes the relationship between ϵ-indistinguishability (a variant of the differential privacy definition) and adversarial privacy, which is of independent interest: an algorithm is ϵ-indistinguishable iff it is private for a particular class of adversaries (defined precisely herein). Our perturbation algorithm guarantees privacy against adversaries in this class whose prior distribution is numerically bounded.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Statistical databases*; G.3 [**Probability and statistics**]: Distribution functions

*Supported in part by the National Science Foundation under the grant IIS-0627585

†Supported in part by the National Science Foundation under the grant IIS-0415193

PODS'09, June 29–July 2, 2009, Providence, Rhode Island, USA.

General Terms

Algorithms, Security, Theory

Keywords

private data analysis, privacy preserving data mining, output perturbation, sensitivity, social networks, join queries

1. INTRODUCTION

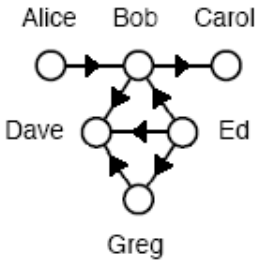
The significant advances in privacy-preserving query answering have focused primarily on datasets describing properties of independent entities, but not relationships. Queries over data containing relationships typically involve joins. For such queries many existing techniques cannot be applied, and others yield severely distorted answers. We are driven to consider data with relationships by the compelling application of social network analysis.

Application: Analysis of private social networks

A social network is a graph representing a set of entities and relationships between them. We model a social network as a simple directed graph, described by a relationship table $R(u, v)$. Figure 1 shows a simple social network, (a), and its relationship table, (b). Each edge of the graph G occurs as a tuple in R.

Social network analysis is a rapidly growing discipline focused on the structure and function of networks. Many of the most interesting social network analyses involve topological properties of the graph, which require self-joins on the relationship table. We list a number of such queries in Fig. 1(c), each of which counts the occurrence in the graph of some pattern or structure. For example, the clustering coefficient [15] of a graph is the likelihood that a friend of one's friend is also a friend. The clustering coefficient can be computed from two of the counting queries shown in the figure: Δ (the number of triangles) and $\vee$ (the number of two-length paths). Analysts also commonly count the occurrence of interesting patterns (sometimes called motif analysis) and the distribution of path lengths in the graph. Queries c_i, $d_{i,v}$, K_i, and $h_{i,j}$ are examples of such queries.

The collection and release of social network data is highly constrained by privacy concerns. Relationships in a social network are often sensitive, as they may represent personal contacts, flows in information, collaborations, employment



(a) Social Network G

Node 1	Node 2
Alice	Bob
Bob	Carol
Bob	Dave
Ed	Bob
Ed	Dave
Greg	Dave
Ed	Greg

(b) Relationship table R

Δ : # of triangles in G
$\vee$: # of 2-length paths in G
c_i : # of cycles of length i in G
$d_{i,v}$: # of nodes at distance i from v
K_i : # of i-node cliques
$h_{i,j}$: # of matches for conn. subgraph h with i nodes and j edges

(c) Social network queries

Figure 1: An example social network represented as a directed graph and edge table, along with selected social network queries.

relationships, etc. Our goal is to permit accurate social network analysis while protecting against the threat of edge disclosure.

Researchers have begun to investigate threats against social networks [1, 11] and a number of anonymization schemes have been proposed [11, 12, 20, 21, 22, 3]. These techniques do not, however, provide quantitative guarantees of privacy and utility. Existing query-answering techniques that do provide guarantees are difficult to adapt to data containing relationships. Rastogi et al. [17] publish a randomized version of a database that supports accurate estimates of counting queries, but does not support joins. Dwork et al. [5] propose an output perturbation mechanism that can be used to answer queries with joins, however the noise added is unacceptably high for some join queries, such as those in Fig. 1(c).

To address the shortcomings of existing techniques we make the following three contributions:

▶ We propose an output perturbation algorithm, similar to the differentially private algorithm of Dwork et al [5]. For a large subclass of conjunctive queries with disequality, we show that the algorithm meets a strong information-theoretic standard of *adversarial privacy* that bounds the worst-case change in the adversary's belief about tuples in the database.

▶ We show that the algorithm significantly improves utility over competing techniques. In particular, for social network queries, such as those in Fig. 1(c), the algorithm improves the error bound from polynomial in the number of nodes to polylogarithmic. This algorithm is (to the best of our knowledge) the first to answer such queries with acceptable accuracy, even for worst-case inputs.

▶ The improved utility is achieved by relaxing the privacy condition. To provide insight into the relaxation in privacy, we characterize the class of adversaries that mechanisms satisfying ϵ-indistinguishability (a variant of the differential privacy definition) protect against. This result is of independent interest, as it describes a fundamental connection between two widely studied notions of differential [5, 4] and adversarial [6, 14, 17, 7] privacy.

We note that all our results apply to arbitrary relational databases, and are not restricted only to social networks. The next section provides an overview of all the main results in the paper. Section 3 provides technical details for the privacy and equivalence results. In Sections 4 and 5 we explain extensions and discuss related work.

2. MAIN RESULTS

In this section we describe our two main results and their significance. We postpone several technical details to Sec. 3. All the proofs are omitted to the full version of the paper [9].

2.1 Background

Each of the social network queries in Fig. 1(c) can be represented as a counting conjunctive query with disequality, expressed in $CQ(\neq)$. In this paper the value of a conjunctive query is defined to be a number, namely the number of satisfying assignments for the variables in the query head. For example, the query

$$\Delta(x, y, z) = R(x, y)R(y, z)R(z, x), x \neq y, y \neq z, z \neq x$$

returns the number of triples x, y, z that form a triangle in the graph.

In the general case, we let $R_1, R_2, \ldots, R_l$ be the relations in the database. We assume that each attribute of the relations takes values from the same domain D and denote $m = |D|$ the size of the domain. In practice one chooses D as the active domain of the database, for instance in Fig 1(b) D is the set of all nodes in the graph, and m the number of such nodes. We denote by $\text{Tup}_i = D^{arity(R_i)}$ the set of all possible tuples in the relation R_i and $\text{Tup} = \cup_i \text{Tup}_i$ the set of all possible tuples in the database. A database instance $I \subseteq \text{Tup}$ is an instantiation of the relations R_i, where $|I|$ is the number of tuples in the database.

2.2 Algorithm

Algorithm 2.1 takes as input the database instance I and a query and returns an output $A(I)$. Our goal is to provide both privacy—$A(I)$ should not disclose specific tuples in I—and utility—$A(I)$ should be a good estimate of the true query answer, $q(I)$. For simplicity, we assume a single query: the generalization to multiple queries is discussed in Sec 4.3. The algorithm also takes as input two privacy parameters γ and ϵ.

On a request for a query q, the algorithm does one of two things: with some probability, it ignores the database and outputs a random number chosen uniformly from the range of q (Step 3). This event occurs with probability p/γ, where p (defined in Step 2) is a function of the query q and m, and is typically small, while γ is the first privacy parameter. Otherwise (Step 4) the algorithm returns the query answer after adding random noise sampled from a Laplace distribution with mean 0 and standard deviation

λ/ϵ, where λ (defined in Step 2) is a function of the query and m, and ϵ is the second privacy parameter.

Algorithm 2.1 Output perturbation (**Inputs:** query q, database I, privacy parameters γ, ϵ)

1: Let g be the number of subgoals in query q and v the number of variables.
2: Set $\lambda = (8(v+1)g^2 \log m)^{g-1}$ and $p = g/m$.
3: With probability p/γ, output $A(I)$ chosen uniformly at random from $\{0, 1, \ldots, m^v\}$.
4: Otherwise (with probability $1 - p/\gamma$), output $A(I)$ as

$$A(I) = q(I) + Lap(\lambda/\epsilon)$$

Dwork et. al. [5] introduced a perturbation algorithm that, for a given query q adds a Laplacian noise λ/ϵ and guarantees ϵ-*indistinguishability*. The λ for that algorithm is determined by the global sensitivity of the query, which roughly denotes the *worst-case* change in the query answer on changing one tuple in any possible database instance.

The key difference in our algorithm is the choice of λ. Returning a completely random response with small probability in Step 3, along with the relaxed notion of adversarial privacy, allows us to choose a much smaller value for λ in Step 4, and as a consequence, our algorithm has much better utility. Roughly speaking, λ for our algorithm is determined by the adversary's prior expectation of the query sensitivity instead of the worst-case sensitivity over all possible database instances. See Sec. 3.2 for the formal details.

For example, given any query in Fig. 1(c) the random noise that our algorithm adds results in quite acceptable utility, while the amount of noise needed to ensure ϵ-indistinguishability renders them useless to a data analyst (we do a detailed comparison in Sec 2.4).

The novelty in our approach relies in the privacy analysis. Instead of guaranteeing ϵ-indistinguishability, we guarantee a somewhat weaker adversarial privacy: while these two paradigms to privacy are quite distinct, and hard to compare, we will show in Sec 2.5 a remarkable connection between the two definitions. Using this connection we are able to claim that our algorithm guarantees adversarial privacy against the same adversaries as ϵ-indistinguishability, as long as they are bounded quantitatively. We explain this below, but first we discuss briefly the utility of the algorithm.

2.2.1 Utility

Our definition of utility is based on how close the output $A(I)$ is to the actual answer $q(I)$.

Definition 2.1 ((Σ, Γ)-usefulness). *An algorithm A is called* (Σ, Γ)-useful *w.r.t query q if for all inputs I:*

$$Pr[\ |q(I) - A(I)| \geq \Sigma\ ] \ \leq \ \frac{\Gamma}{m}$$

Here the probability Pr is computed solely on the random coin tosses of the algorithm A

We prove in the full version [9] the following theorem.

Theorem 2.2 (Utility). *Let q be any query with g subgoals. Then Algorithm 2.1 is (Σ, Γ)-useful w.r.t q for $\Sigma = O(\log^g m/\epsilon)$ and $\Gamma = O(1/\gamma)$.*

Example Let us fix the privacy parameters ϵ and γ as constants independent of m. Consider again the query Δ that counts the number of triangles in R. Then, g, the number of subgoals of Δ is 3 and hence the Algorithm 2.1 guarantees (Σ, Γ)-usefulness for $\Sigma = O(\log^3 m)$ and $\Gamma = O(1)$. This means that the algorithm outputs a response with error $O(\log^3 m)$, except with probability $O(1/m)$. Recall that m is the number of nodes in R.

2.3 Adversarial Privacy

In the adversarial definition of privacy an *adversary* is a probability distribution: it defines the adversary's prior belief. An algorithm is private if the a posteriori distribution (after seeing the algorithm's output) is close to the prior. Moreover, this property should hold for any reasonable adversary.

In our setting, the adversary's prior knowledge is a probability distribution P on database instances. We denote $P(I)$ the probability of an instance I, and $P(t)$ marginal probability of a tuple, $P(t) = \sum_{I:t\in I} P(I)$. After seeing the output $O = A(I)$ of the algorithm, the adversary changes his belief to the a posteriori $P(t \mid O)$. Privacy means that the a posteriori should be close to the prior.

Since we do not know P (only the attacker knows it) we consider an entire class of distributions $\mathcal{P}$: the adversary can be any $P \in \mathcal{P}$.

Definition 2.3 (Adversarial Privacy). *An algorithm A is (ϵ, γ)-adversarially private w.r.t. $\mathcal{P}$ iff for all outputs O of A, tuples $t \in Tup$ and prior distributions $P \in \mathcal{P}$ the following holds:*

$$P(t \mid O) \leq e^{\epsilon} P(t) + \gamma$$

Note that for a fixed O, the probability $P(t|O)$ depends on the prior P and the definition of A, but not on the coin tosses (randomness) of any particular run of A. We emphasize that the privacy guarantee is given for every output O and thus for every input instance I, and is not an average-case guarantee.

If an algorithm is $(\epsilon, 0)$-adversarially private, then we simply say that it is ϵ-adversarially private. Related definitions in the literature include (ρ_1, ρ_2) breach [6], (d, γ)-privacy [17], and Safe$_\Pi$ [7]. Like the latter, we do not protect against negative leakages (but see Sec. 4.3), which is acceptable in many, but not all applications.

The main difficulty in applying adversarial privacy is choosing the class of adversaries $\mathcal{P}$. If one chooses $\mathcal{P}$ to be the set of all possible adversaries, then no algorithm can be both useful and adversarially private. Paraphrasing an example from Dwork [4], suppose a malicious adversary doesn't know how many friends Alice has, but knows that Alice has a number of friends that is exactly 0.01% of the total number of triangles in the data. Then, if an algorithm returns the total number of triangles, a significant amount of information has been leaked. We cannot hope to protect adversarial privacy against such an omnipotent adversary, and the common practice is for a security manager to assume a weaker threat. The ability to adjust the threat model, hence the class of adversaries $\mathcal{P}$, is of key importance in practice, yet no principled way to define the class $\mathcal{P}$ has been proposed in the literature.

In this paper we propose a principled definition of the power of the adversary, as quantitative restrictions of the

class $\mathcal{PTLM}$ (defined formally in Sec. 2.5). The importance of the class $\mathcal{PTLM}$ is that adversarial privacy with respect to this class is precisely ϵ-*indistinguishability* [5] (reviewed in Sec. 2.5). In other words, ϵ-indistinguishability can be described precisely in terms of a particular class of adversaries, namely $\mathcal{PTLM}$ adversaries, which justifies our choice. Once we have this yardstick, we introduce the following quantitative restrictions on adversaries:

DEFINITION 2.4 ((δ, k)-BOUNDED). *Let $\delta \in [0, 1]$ and $k \in \mathbb{N}$. An adversary P is (δ, k)-bounded if (i) At most k tuples have marginal probability $P[t] = 1$, and (ii) For all other tuples $t \in Tup$ the marginal probability $P[t]$ is less than δ.*

Algorithm 2.1 guarantees privacy for all queries in a certain subset of $CQ(\neq)$, which we call *stable queries*. We postpone the formal definition of stable queries until Sec. 3.1, but show here that the following important class of queries is stable.

PROPOSITION 2.5 (SUBGRAPH QUERY STABILITY). *Let q be the query that counts the number of occurrences of a subgraph h in G. If h is connected (in the traditional graph-theoretic sense), then q is stable.*

All queries in Fig. 1(c) with the exception of $d_{i,v}$ correspond to queries that count connected subgraphs, and hence are stable. The class of stable queries is in fact much more general and includes the query $d_{i,v}$ as well. We can now state our first main result (the proof technique is in Sec. 3.2):

THEOREM 2.6 (PRIVACY). *Let $\delta = \log m/m$ and $k = \log m$ be the parameters bounding the adversary's power. Then if the input query q is stable, Algorithm 2.1 ensures (ϵ, γ)-adversarial privacy w.r.t any (δ, k)-bounded $\mathcal{PTLM}$ adversary.*

Finally, we comment on choosing the parameters for bounding the adversary quantitatively. In many applications, assuming a (δ, k) bound on the adversary is quite reasonable. For example, in online social networking applications, users can only access a limited subgraph of the actual social network. Facebook allows users to know only their friends, and the friends of their friends. Thus, a bound of $k = \log m$ is a reasonable assumption on the power of the adversary. Another example is the *active attackers* considered in [1] that can create an arbitrary subgraph of $k = O(\sqrt{\log m})$ edges. For edges other than the k known edges, we assume an upper bound of $\delta = \log m/m$, which is also appropriate for sparse networks observed in practice in social networks [15]. The value is not critical for the functioning of our algorithm and we consider larger δ in Sec. 4.3.

2.4 Discussion

To put our results in perspective, we compare Algorithm 2.1 to the algorithm of Dwork et al. [5] (call it DMNS) in terms of both privacy and utility.

These algorithms are modeled on different notions of privacy, but our equivalence result (Section 2.5) allows us to relate them more easily. DMNS is ϵ-indistinguishable, which is equivalent to $(\epsilon, 0)$-adversarial privacy w.r.t. any adversary in the $\mathcal{PTLM}$ class. In contrast, Algorithm 2.1 only protects against a subclass of $\mathcal{PTLM}$, namely those that are (δ, k)-bounded. It also tolerates slightly more disclosure: it is (ϵ, γ)-adversarially private where $\gamma > 0$. To compare utility, we fix ϵ to be the same and γ some positive constant.

In terms of utility, both algorithms offer guarantees that depend on the particular query. We compare the social network queries in Figure 1(c) in terms of (Σ, Γ)-usefulness. We fix $\Gamma = O(1)$—thus, both algorithms have a $O(1/m)$ probability of returning a useless output—and compare Σ, the bound on error. For Algorithm 2.1, the bound on the error follows from Theorem 2.2. For DMNS, the bound comes from the fact that its error is distributed exponentially with mean GS_q/ϵ, where GS_q is the global sensitivity of q. Global sensitivity is formally defined in Section 3.1, but intuitively it is the worst-case change in the query answer given the change of a single tuple in any possible database instance.

Table 1 compares the error bounds of the two algorithms. As additional context, it also shows a lower bound on the error of an optimal ϵ-indistinguishable algorithm, denoted OPT_{ind}—this bound is derived in the full version [9]. In order to achieve ϵ-indistinguishability for the example queries, one needs to introduce error rates that are linear or polynomial in m (the number of nodes in the graph), which makes them useless for social network analysts. By contrast our algorithm guarantees error bounds that are polylogarithmic in m. (The $d_{i,v}$ query has a range of $\{0, \ldots, m\}$, so the only practically useful algorithm is Algorithm 2.1 for small i. For K_i, Algorithm 2.1 is useful for small i, but no algorithm is useful for large i.)

Table 1: The error bound (Σ) of Algorithm 2.1 compared with ϵ-indistinguishable alternatives for the queries of Figure 1(c). Recall that m = the number of nodes in the graph.

	Σ		
Query	OPT_{ind}	DMNS [5]	Algorithm 2.1
Δ	$\Omega(m)$	$\Theta(m \log m/\epsilon)$	$\Theta(\log^3 m/\epsilon)$
$\vee$	$\Omega(m)$	$\Theta(m \log m/\epsilon)$	$\Theta(\log^2 m/\epsilon)$
c_i	$\Omega(m^{i-2})$	$\Theta(m^{i-2} \log m/\epsilon)$	$\Theta(\log^i m/\epsilon)$
$d_{i,v}$	$\Omega(m)$	$\Theta(m \log m/\epsilon)$	$\Theta(\log^i m/\epsilon)$
K_i	$\Omega(m^{i-2})$	$\Theta(m^{i-2} \log m/\epsilon)$	$\Theta(\log^{i^2} m/\epsilon)$
$h_{i,j}$	$\Omega(m^{i-2})$	$\Theta(m^{i-2} \log m/\epsilon)$	$\Theta(\log^j m/\epsilon)$

Note that all of these queries have joins. If a query does not have joins—such as *count the number of edges in the graph*—it has low global sensitivity and DMNS has a similar error rate (but better constants). However, these examples illustrate that joins can lead to high global sensitivity and Algorithm 2.1 can guarantee much more accurate answers than a ϵ-indistinguishable algorithm.

Queries with high global sensitivity were also considered by Nissim et al. [16], who propose adding noise proportional to a smooth upper bound on the *local* sensitivity (i.e., sensitivity of the instance rather than worst-case global sensitivity). A direct comparison with our approach is difficult because while the smooth bounds were derived for a few specific queries (e.g., median), it is not known how to compute them in general, nor in particular, for the social network queries we consider here. (See Section 5.2 for further discussion.)

2.5 Adversarial Privacy vs. Indistinguishability

In this section, we state our second main result, which states that ϵ-indistinguishability is the same as ϵ-adversarial privacy w.r.t. $\mathcal{PTLM}$ adversaries. Indistinguishability was defined in [5] and we review it briefly. Consider a randomized algorithm A that, on a given input instance I, returns a randomized answer $A(I)$. Given an answer O, $\Pr(A(I) = O)$ denotes the probability (over the random coin tosses of A) that the algorithm returns exactly O on input I.

For simplicity, we denote $I + t$ the instance obtained by adding a tuple t which is not in I to the set I.

DEFINITION 2.7 (ϵ-INDISTINGUISHABILITY). *An algorithm A is ϵ-indistinguishable iff forall instances I and I' that that differ on a single tuple (i.e. $|I| = |I'|$ and there exist tuples t, t' such that $I + t = I' + t'$), and for any output O, the following holds:*

$$Pr[A(I') = O] \leq e^{\epsilon} Pr[A(I) = O] \quad (1)$$

A related notion is that of ϵ-differential privacy [4], which guarantees condition (1) forall I and I' such that one contains an additional tuple compared to the other (i.e. there exists a tuple t such that either $I + t = I'$ or $I' + t = I$). The distinction between these two notions of privacy is sometimes blurred in the literature and the term differential privacy is used to denote indistinguishability. However, we use the term indistinguishability throughout the paper, since this is the precise definition for which our main results apply.

Indistinguishability does not mention an adversary, but was shown to protect against *informed* adversaries [5], who know all but one tuple in the database. However, the *informed* adversaries do not have tuple correlations. Against adversaries with arbitrary correlations (such as "Alice's height is 5 inches shorter than the average" [4]), ϵ-indistinguishability does not guarantee adversarial privacy. An important question is: for which adversaries does ϵ-indistinguishability guarantee adversarial privacy?

Our result shows that ϵ-indistinguishability corresponds precisely to ϵ-adversarial privacy for a certain $\mathcal{PTLM}$ class. The class $\mathcal{PTLM}$ has never been defined before and requires us to provide the following background on log-submodular and planar distributions.

While well known in probability theory [2], log-submodular ($\mathcal{LM}$) distributions have been considered only recently in the context of data privacy in [7].

DEFINITION 2.8 (LOG-SUBMODULARITY). *A probability distribution P is log-submodular (denoted $\mathcal{LM}$) if forall instances I, I' the following condition holds:*

$$P(I)P(I') \geq P(I \cap I')P(I \cup I') \quad (2)$$

To understand the intuition behind a log-submodular distribution, recall that a distribution P is called *tuple-independent* if for any two tuples t_1, t_2, $P(t_1, t_2) = P(t_1)P(t_2)$; we write $\mathcal{IND}$ for the class of all tuple-independent distributions. One can check that P is tuple independent iff for any two instances[1] $P(I)P(I') = P(I \cap I')P(I \cup I')$. Hence, $\mathcal{IND} \subseteq \mathcal{LM}$. Most prior work on adversarial data privacy, including our own [14, 17] has considered only tuple-independent adversaries. The $\mathcal{LM}$ adversaries, considered first in [7], are more powerful since they allow correlations. However, they turn out to be too powerful to capture ϵ-indistinguishability because, as we explain next, they are not closed under marginalization.

[1]By direct expansion all four probabilities, e.g. $\Pr(I) = \prod_{t \in I} P(t) \cdot \prod_{t \notin I}(1 - P(t))$, etc.

Marginalization Let $Q \subseteq Tup$ be any set of tuples. Given an algorithm A, its *marginalization* w.r.t. Q is $A_Q(I) = A(I \cap Q)$. Thus, A_Q just ignores the tuples outside the set Q. One can easily check that if A is ϵ-indistinguishable and Q is a fixed set, then A_Q is also ϵ-indistinguishable. Thus, if ϵ-indistinguishability is to be captured by some class of adversaries, that class must be closed under the operation of marginalization, defined as follows:

DEFINITION 2.9 (MARGINAL DISTRIBUTION). *Let P be any probability distribution over possible databases. Then the marginal distribution over Q, denoted as P_Q, is a probability distribution over the subsets S of Q defined as $P_Q(S) = \sum_{I: I \cap Q = S} P(I)$.*

$\mathcal{LM}$ distributions are not closed under marginalization, hence they cannot capture ϵ-indistinguishability. This justifies our definition of *total* log-submodular distributions.

DEFINITION 2.10 (TOTAL LOG-SUBMODULARITY). *A probability distribution P is* total log-submodular *($\mathcal{TLM}$) if for every set $Q \subseteq Tup$, the marginal distribution P_Q is log-submodular. Formally:*

$$\forall Q,\ \forall S, S' \subseteq Q,\ P_Q(S)P_Q(S') \geq P_Q(S \cap S')P_Q(S \cup S')$$

The $\mathcal{TLM}$ class contains $\mathcal{IND}$ and all distributions with negative correlations. Intuitively, by negative correlations we mean that $P(t_1|t_2) \leq P(t_1)$ for all tuples t_1, t_2. We formalize the notion of negative correlations in Sec. 3.3.

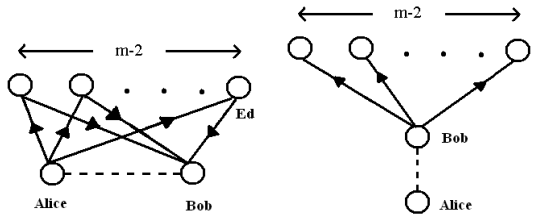
Planarity Finally, our last definition is motivated by the following observation. It is implicit in the definition of ϵ-indistinguishability that the size of the database is made public. To see that, consider the following algorithm A: given an instance I, the algorithm returns the cardinality of I. A is trivially ϵ-indistinguishable, for every $\epsilon > 0$. In terms of adversarial privacy, this means that the adversary knows the size of the database. The corresponding notion in probability theory is called *planarity* [18].

DEFINITION 2.11 (PLANAR DISTRIBUTIONS). *A distribution P is* planar *if there exists a number n such that forall I if $|I| \neq n$ then $P(I) = 0$. We call n the "size of the database" for the distribution P.*

We denote by $\mathcal{PTLM}$ the class of planar, $\mathcal{TLM}$ adversaries. At last, we show our second main result: a complete equivalence of ϵ-indistinguishability and adversarial privacy w.r.t $\mathcal{PTLM}$ adversaries. The proof technique is in Sec. 3.3.

THEOREM 2.12 (EQUIVALENCE). *Let A be any algorithm and $\epsilon > 0$ be any parameter. Then the following two statements are equivalent (i) A is ϵ-indistinguishable, (ii) A is ϵ-adversarially private w.r.t $\mathcal{PTLM}$ adversaries.*

This equivalence result has interesting implications for both privacy definitions. In Sec 4, we propose a relaxation of ϵ-indistinguishability and also show how adversarial privacy provides protection of boolean properties and resistance to some positive correlations.



2(a) Triangles **2(b)** length 2 paths

Figure 2: Graphs showing high sensitivity

3. TECHNICAL RESULTS

In this section, we define and characterize the class of stable queries, which are queries that Algorithm 2.1 protects privacy for. We also explain the technical results behind our two main results: the privacy Theorem 2.6 and the equivalence result (Theorem 2.12).

3.1 Stable Queries

The intuition behind the definition of stable queries is based on the the notion of local sensitivity. This notion was defined in [16] and denotes the maximum change in the query answer on changing any one tuple of the particular input instance.

DEFINITION 3.1 (LS_q [16]). *The local sensitivity of a query q on instance I is the smallest number $LS_q(I)$ such that for all database instances I' that differ on a single tuple with I (i.e. $|I| = |I'|$ and there exist tuples t, t' such that $I + t = I' + t'$),*

$$|q(I) - q(I')| \leq LS_q(I)$$

Global sensitivity [5] is the worst-case local sensitivity over all possible instances on the domain. Dwork et al. [5] propose an algorithm that guarantees good utility for all queries with low global sensitivity.

The queries listed in Fig 1(c), however, have high global sensitivity. For instance, the global sensitivity of Δ defined in Figure 1(c) is $m-2$, where m is the number of nodes in G. This is illustrated by the graph shown in Figure 2(a). Suppose the graph contains the edge $(Alice, Bob)$, but does not contain the edge $(Bob, Alice)$. Then the number of triangles in the graph is 0. If we change the edge $(Alice, Bob)$ to the edge $(Bob, Alice)$, the number of triangles becomes $m-2$, showing that the sensitivity of Δ is $m-2$. Similarly, Figure 2(b) illustrates the high global sensitivity of $\vee$.

Stable queries are those that have low local sensitivity in *expectation*, where the expectation is computed over database instances drawn from adversary's prior distribution. Since we do not know the adversary's prior, we give a syntactic criteria for queries that guarantee low expected local sensitivity based only on the quantitative bounds δ, k of the adversary (see Def. 2.4). We define two notions: *dense queries*, which are queries that have low expected value w.r.t. adversary's prior distribution, and the *derivative set* of a query q, which contains queries that upper bound the local sensitivity of q. Roughly speaking, a stable query is a query whose *derivative set* contains only *dense* queries.

Dense queries Denote $goals(q)$ the set of subgoals in the query q. Also denote $vars(q)$ ($const(q)$) the set of variables (constants) in q. We define the density of q the ratio $\frac{|goals(q)|}{|vars(q)|}$. We say a query is dense if its density and the density of all of its possible *homomorphic images* (defined below) is greater than one.

DEFINITION 3.2 (HOMOMORPHISM). *A homomorphism is a function $h : vars(q) \rightarrow vars(q) \cup const(q)$ that maps each variable in $vars(q)$ to either a variable in $vars(q)$ or a constant in $const(q)$ while ensuring that the inequality constraints of the query are satisfied.*

For a homomorphism h, denote q_h the query obtained by replacing, for all variables $x \in vars(q)$, each occurrence of x in q by the symbol $h(x)$. q_h is called the h-homomorphic image of q. We can now define dense queries formally.

DEFINITION 3.3 (DENSE QUERY). *Let H denote the set of all homomorphisms for a query q. We say q is dense if all the homomorphic images of q (i.e. queries in the set $\{q_h | h \in H\}$) have density ≥ 1.*

Examples The query Δ of Fig 1(c) is dense. It has a density of 1 and all of its homomorphic images have higher densities. The query $\vee$, defined as $R(x,y)R(y,z), x \neq y, y \neq z, z \neq x$, is not dense as its density is 2/3. The query $q(x,y,z) : -R(x,y,z)R(x,z,y)R(x,y,y)$ is not dense, even though its density is 1. This is because its homomorphic image $q_h(x,z) : -R(x,z,z)$ (for $h(y) = z$) has density 1/2.

Derivative Now we define the notion of derivative used for stable queries. Define a set of new constants *dummy* with $|dummy| = |vars(q)|$ and the set $D_0 = dummy \cup const(q)$. Let $\mathbf{x} = \{x_1, \ldots, x_l\}$ be any l variables of q and $\mathbf{a} = \{a_1, \ldots, a_l\}$ be any l constants from D_0. Denote $q[\mathbf{x}/\mathbf{a}]$ the query obtained by substituting all occurrences of variable x_i in q by the constant a_i for all $i = \{1, \ldots, l\}$.

DEFINITION 3.4 (DERIVATIVE). *Let g be a subgoal of q with l variables $\mathbf{x} = \{x_1, \ldots, x_l\}$ and let $\mathbf{a} = \{a_1, \ldots, a_l\}$ be a set of l constants. The derivative of q w.r.t. $(g, \mathbf{a})$, denoted as $\frac{\partial q}{\partial(g,\mathbf{a})}$, is the query obtained by removing the subgoal g from the query $q[\mathbf{x}/\mathbf{a}]$.*

Intuitively, the query $\frac{\partial q}{\partial(g,\mathbf{a})}$ represents the change in the answer to q if the tuple $\mathbf{a} = \{a_1, \ldots, a_l\}$ is added in the relation g. For instance, $\frac{\partial \Delta}{\partial(R(x,y),(a,b))}$ is the query $R(b,z)R(z,a)$, which measures the change in the number of triangles on adding the edge $R(a,b)$. Next we define the derivative set which contains all possible derivatives of the query.

DEFINITION 3.5 (DERIVATIVE SET). *The derivative set of q, denoted as $\partial^1(q)$, is the set of queries $\{\frac{\partial q}{\partial(g,\mathbf{a})} | g \in goals(q), \mathbf{a} \in D_0^{|vars(g)|}\}$.*

Due to technical reasons, we also need to define higher order derivate sets. The i^{th} order derivative set of q, denoted as $\partial^i(q)$, is recursively defined as the set of derivate queries of the $(i-1)^{th}$ order derivative set. That is $\partial^i(q) = \cup_{q' \in \partial^{i-1}(q)} \partial^1(q')$. We can now define stable queries.

DEFINITION 3.6 (STABLE). *A query q is stable if for all $i \geq 1$ and queries $q' \in \partial^i(q)$, q' is dense.*

Examples Any single subgoal query is automatically stable. Proposition 2.5 proves the stability of a large class of queries. $d_{i,v}$ defined as $q(x_i) : -R(v,x_1)R(x_1,x_2)\ldots, R(x_{i-1},x_i)$, is also stable. This is because all derivatives of $d_{i,v}$ are dense.

Checking for stable queries The query complexity for checking whether a query is stable is in coNP. To see this, note that a witness for showing a query is not stable is simply a derivative query and a homomorphism for the derivative. Checking whether the homomorphic image of the derivative has density < 1 can then be done in polynomial time.

Despite the high query complexity, stability is a property of the query and does not depend on the data. This is an advantage compared to some techniques, such as the approach of Nissim et al. [16], that must look at the data to determine how to answer the query. (See Section 5 for more discussion of related work.)

3.2 Proof of Privacy (Theorem 2.6)

The proof relies on the property that the local sensitivity of a query is bounded w.h.p. according to the adversary's prior distribution. A similar notion was also considered in [10]. Denote $P(LS_q > \lambda) = \sum_{I:LS_q(I)>\lambda} P(I)$, the probability that the local sensitivity of q is greater than λ according to adversary's prior distribution P. Next we show the value of λ that makes this probability small for all stable queries.

PROPOSITION 3.7 (RESULT-1). *Let $\delta = \log m/m$ and $k = \log m$. Let P be any (δ, k)-bounded $\mathcal{PTLM}$ adversary and q be a stable query with g subgoals and v variables. Then for $\lambda = 8(v+1)g^2 \log m$, we have*

$$P(LS_q > \lambda) \leq \frac{g}{m^{v+1}}$$

The proof of Proposition 3.7 is in the full version [9]. It uses new and non-trivial concentration results, which are also discussed in the full version. The concentration results extend those shown by Vu et. al. [19] for tuple-independent distributions to $\mathcal{PTLM}$ distributions. These results may be of independent interest as they work for a general class of queries and probability distributions.

Next we show the following privacy result for Algorithm 2.1. Its proof appears in the full version and exploits the connection between ϵ-indistinguishability and adversarial privacy w.r.t. $\mathcal{PTLM}$ adversaries.

PROPOSITION 3.8 (RESULT-2). *Let P be any $\mathcal{PTLM}$ adversary and let q be any query with v variables. Let ϵ, γ be the privacy parameters and let λ, p be the parameters used in Steps 3 and 4 of Algorithm 2.1. If $P(LS_q > \lambda) \leq \frac{p}{m^v}$, then Algorithm 2.1 guarantees (ϵ, γ)-adversarial privacy w.r.t. P while answering q.*

Result-1 (Prop. 3.7) shows that if P is a (δ, k)-bounded $\mathcal{PTLM}$ adversary, then any stable query q satisfies $P(LS_q > \lambda) \leq \frac{p}{m^v}$ for $\lambda = 8(v+1)g^2 \log m$ and $p = g/m$. Now these are the values that have been used in steps 3 and 4 of the Algorithm 2.1. Thus we can apply Result-2 (Prop. 3.8) on the adversary P and query q and obtain that (ϵ, γ)-adversarial privacy w.r.t P is preserved while answering q by Algorithm 2.1. Since this holds for any (δ, k)-bounded $\mathcal{PTLM}$ P and stable query q, we get the privacy result of Theorem 2.6.

3.3 Proof of the Equivalence (Theorem 2.12)

In this section, we explain the technique behind the proof of Theorem 2.12. We treat each tuple $t \in Tup$ as a boolean variable, which is true iff t belongs to the actual database instance. We also define boolean formulas over these boolean variables. For example the formula $(t_1 \vee t_2)$ states that either t_1 or t_2 belongs to the database. We denote $vars(b)$ the set of variables that occur in a formula b. A formula m is *monotone* if switching any variable t from false to true does not change its value from true to false. For example, the formula $m = t_1 \vee t_2$ is monotone.

Our proof proceeds by establishing a connection between the $\mathcal{PTLM}$ class and a class of distributions, called $\mathcal{CNA}$, which only have negative correlations among tuples. Many notions of negative correlations exist in the literature and our proof exploits the relationship among them. We discuss them in the full version [9]. Here we just define the most important one.

DEFINITION 3.9 ($\mathcal{NA}$). *A distribution P is said to be negatively associated ($\mathcal{NA}$) if for all tuples t and monotone formulas m s.t. $t \notin vars(m)$, $P[t \wedge m] \leq P[t]P[m]$.*

For a distribution P, denote P_t the distribution obtained by conditioning P on tuple t (i.e. for all I, $P_t(I) = P(I|t)$). Similarly define $P_{\neg t}$. A class of distributions $\mathcal{P}$ is said to be closed under conditioning if for all distributions $P \in \mathcal{P}$ and all tuples t, the distributions P_t and $P_{\neg t}$ also belong to $\mathcal{P}$. One can show that the $\mathcal{PTLM}$ class is closed under conditioning. Define $\mathcal{CNA}$ to be the largest subclass of $\mathcal{NA}$ distributions that is closed under conditioning. Next we show an equivalence relationship between $\mathcal{PTLM}$ and $\mathcal{CNA}$.

PROPOSITION 3.10 (NEG. CORRELATIONS). *Let P be a planar distribution. Then the following two statements are equivalent. (i) P is $\mathcal{CNA}$, (ii) P is $\mathcal{PTLM}$*

The proof of the proposition appears in the full version. The proof is based on the main lemma (lemma 3.2) of [8], which establishes a connection between *balanced matroids* (see [8]) and $\mathcal{CNA}$ distributions. To complete the proof of Theorem 2.12, it suffices to prove the following proposition.

PROPOSITION 3.11 (EQUIVALENCE-2). *Let A be any algorithm and $\epsilon > 0$ be any parameter. Then the following statements are equivalent (i) A is ϵ-indistinguishable, (ii) A is ϵ-adversarially private w.r.t. planar $\mathcal{CNA}$ adversaries.*

The proof appears in the full version. We give here a brief comment. The proof of $(ii) \Rightarrow (i)$ is simple and works by considering an extreme adversary P who knows all but one tuple in the database. The proof of $(i) \Rightarrow (ii)$ is harder and uses an expansion property of balanced matroids.

Further we conjuncture that the planar $\mathcal{CNA}$ class is maximal among all classes for which ϵ-indistinguishability is ϵ-adversarial privacy. For this we would ideally like to prove that for every adversary P that is not planar $\mathcal{CNA}$, there exists an ϵ-indistinguishable algorithm that is not ϵ-adversarially private w.r.t P. Since we restrict ourselves to classes that are planar and closed under conditioning, we would like to prove this result for any adversary P which is not $\mathcal{NA}$.

Next we state a result (proof is in the full version) that makes progress in this direction. We consider an adversary P which is not $\mathcal{NA}$, and thus $P[t \wedge m] \geq P[t]P[m]$ for some tuple t and monotone formula m. However we further assume that $P[t \wedge m] \geq P[t]P[m] + P[t]^2$. With this additional assumption on P, we show the existence of an ϵ-indistinguishable algorithm that is not ϵ-adversarially private w.r.t P.

THEOREM 3.12 (MAXIMAL). *Let P be any adversary. Suppose there exists a tuple t and monotone formula m such that $t \notin vars(m)$ and $P[t \wedge m] \geq P[t]P[m] + P[t]^2$. Then there exists an $\epsilon > 0$ and an ϵ-indistinguishable algorithm that is not ϵ-adversarially private w.r.t P.*

4. EXTENSIONS

As shown in Section 2.5, ϵ-indistinguishability is precisely adversarial privacy w.r.t $\mathcal{PTLM}$ adversaries. This result has some interesting implications and we discuss two in this section. We then discuss some extensions to Algorithm 2.1.

4.1 Relaxation of Indistinguishability

From the equivalence result of theorem 2.12, we know that ϵ-indistinguishability implies adversarial privacy w.r.t $\mathcal{PTLM}$ adversaries. Below we give a sufficient condition that implies adversarial privacy w.r.t (δ, k)-bounded $\mathcal{PTLM}$ adversaries. This sufficient condition is expressed as a property of the algorithm, similar to ϵ-indistinguishability, and can be thought of as its relaxation. We believe that this relaxation can support algorithms with better utility than ϵ-indistinguishability, especially in the input perturbation setting where perturbed data is to be published.

To give the relaxation, we define the notion of ϵ-distinguishable tuples. Let t, t' be two tuples. Given an instance I that contains t but not t', denote $I(t, t')$ the instance obtained by replacing t in I by t'.

DEFINITION 4.1 (ϵ-DISTINGUISHABLE TUPLE). *An algorithm A makes a tuple t' ϵ-distinguishable w.r.t tuple t if there exist an instance I that contains t but not t', and an output O of the algorithm A for which $Pr[O|I] \geq e^{\epsilon} Pr[O|I(t, t')]$.*

Thus for an algorithm A that is not ϵ-indistinguishable, ϵ-distinguishable tuples are those for which A fails to guarantee ϵ-indistinguishability.

DEFINITION 4.2 (RELAXED INDISTINGUISHABILITY). *Let l be any positive integer. An algorithm A is (l, e^{ϵ})-relaxed indistinguishable if for every tuple t there are at most l ϵ-distinguishable tuples.*

If $l = 0$, then the relaxed definition above is same as ϵ-indistinguishability. If $l = |Tup|$, then the relaxed definition gives no privacy guarantee. Larger the l, the weaker is the relaxed definition.

THEOREM 4.3 (BOUNDED ADVERSARIAL PRIVACY). *Let $0 < r < 1$ be any number and let A be $(\frac{1-\delta}{\delta}(1-r) - 1, re^{\epsilon})$-relaxed indistinguishable. Then A is also ϵ-adversarially private w.r.t (δ, k)-bounded $\mathcal{PTLM}$ adversaries.*

Theorem 4.3, proved in the full version [9], gives a justification for relaxed indistinguishability. It states that the relaxation is sufficient to guarantee adversarial privacy w.r.t (δ, k)-bounded $\mathcal{PTLM}$ adversaries. Furthermore, since adversarial privacy is a worst-case guarantee made on all possible inputs I and algorithm's outputs, so is this relaxation. Thus it is different from the average-case relaxations considered in the past (such as (ϵ, δ)-semantic privacy [10]).

4.2 Some Properties of Adversarial Privacy

We mention here some properties of adversarial privacy that follow from its relationship with ϵ-indistinguishability.

4.2.1 Protecting Boolean Properties

The adversarial privacy definition 2.3 is stated to protect against tuple disclosures. However, in some cases, we would like to protect general boolean properties such as "Does `John` have a social security number (ssn) starting with the digits 541?". We explain here how adversarial privacy protects all monotone boolean properties[2], which as defined in Sec. 3.3 are boolean formulas with only positive dependence on tuples.

For example, consider the relation $R(\texttt{name}, \texttt{ssn})$. Suppose a $\mathcal{PTLM}$ adversary P is interested in the `ssn` of `John` and knows that one tuple of the form $R(\texttt{John}, *)$ exists in the database. Then adversarial privacy w.r.t P implies that the posterior of P will be bounded for a property like $R(\texttt{John}, \texttt{541}*)$ discussed above. This, in fact, holds for any monotone boolean property over the set of tuples of the form $R(\texttt{John}, *)$. Formally, we can show the following.

PROPOSITION 4.4 (BOOLEAN PROPERTIES). *Let A be any ϵ-adversarially private algorithm w.r.t $\mathcal{PTLM}$. Let T be any set of tuples and P be any $\mathcal{PTLM}$ adversary that knows exactly one tuple in T belongs to the database ($P(|T \cap I| = 1) = 1$). Then for any monotone boolean property m over tuples in T and for all outputs O of A, $P(m|O) \leq e^{\epsilon} P(m)$.*

4.2.2 Protecting against Positive Correlations

As mentioned in Sec. 2.5, the $\mathcal{PTLM}$ class contains distributions with only negative correlations among tuples. Thus adversarial privacy w.r.t $\mathcal{PTLM}$ seems to protect against adversaries with only negative correlations.

However, we can show that protection against negative correlations automatically guarantees protection against adversaries with limited positive correlations albeit with a weaker privacy parameter. We will define below a class $\mathcal{PTLM}^k$ that contains distributions for which any tuple can have positive correlations with at most $k-1$ other tuples, but should have negative correlations with all other tuples. Then we shall show that ϵ-adversarial privacy w.r.t $\mathcal{PTLM}$ adversaries implies $k\epsilon$-adversarial privacy w.r.t $\mathcal{PTLM}^k$ adversaries. To define $\mathcal{PTLM}^k$, we first need to consider the following superclass of $\mathcal{LM}$ distributions.

DEFINITION 4.5 ($\mathcal{LM}^k$). *A probability distribution P is $\mathcal{LM}^k$ if forall instances I, I' such that $|I \setminus I'|$ and $|I \setminus I'|$ are both greater than k the following condition holds:*

$$P(I)P(I') \geq P(I \cap I')P(I \cup I')$$

Thus $\mathcal{LM}^1$ is precisely the $\mathcal{LM}$ class. Further $\mathcal{LM}^{k-1} \subseteq \mathcal{LM}^k$ for all k. Now similar to $\mathcal{TLM}$ and $\mathcal{PTLM}$ define the classes: $\mathcal{TLM}^k$ to be the set of distributions which are $\mathcal{LM}^K$ under every possible marginalization, and $\mathcal{PTLM}^k$ to be the set of planar $\mathcal{TLM}^k$ distribution. Obviously, $\mathcal{PTLM}^{k-1} \subseteq \mathcal{PTLM}^k$ for all k and $\mathcal{PTLM}^1 = \mathcal{PTLM}$. Next we show the following result.

PROPOSITION 4.6 (POSITIVE CORRELATIONS). *Let A be any ϵ-adversarially private algorithm w.r.t $\mathcal{PTLM}$. Then A is $k\epsilon$-adversarially private algorithm w.r.t $\mathcal{PTLM}^k$ adversaries.*

[2]The monotonicity condition is because adversarial privacy protects against only positive leakage and allows negative leakage.

Note that the parameter ϵ appears as e^ϵ in the privacy definition (see Def. 2.3). Thus $k\epsilon$ is actually exponentially worse than ϵ in terms of the privacy guarantee, and Proposition 4.6 makes practical sense only if we apply it for small k. The proof of Proposition 4.6 appears in the full version and exploits the collusion resistance behavior of ϵ-indistinguishability.

4.3 Extensions to Algorithm 2.1

Multiple queries Algorithm 2.1 as stated before can answer only a single query. Now we show how it can be extended to answer a sequence of queries. Consider a sequence of input queries $q_1, \ldots, q_l$. As in Step 2 of the algorithm 2.1, we compute the parameters λ and p. The parameter p is computed as before. However, a separate λ_i is computed for each q_i as $(8g_i^2(v+1)\log m)^{g_i-1}$. Here g_i is the number of subgoals in q_i and v is the total number of variables in all queries. After this the algorithm is same as algorithm 2.1. It decides with probability p/γ to ignore the database and give a completely random answer to each query. Otherwise, with probability $1-p/\gamma$, it returns $A_i(I) = q_i(I) + Lap(\lambda_i/\epsilon)$ for each query q_i.

The algorithm has the same privacy guarantee, but a slightly worse utility guarantee. The algorithm guarantees (Σ_i, Γ_i)-usefulness for $\Sigma_i = O(l \log^{g_i} m/\epsilon)$. This Σ_i is l times worse than the Σ guaranteed in theorem 2.2, where l is the number of queries answered.

Larger δ In section 2, we chose $\delta = \log m/m$. Here we note that a larger δ with value $m^\alpha(\frac{\log m}{m})$ can be used while guarantying the same privacy and the same utility for any query q. The α can be as large as $1 - density(q_h)$, where q_h is the homomorphic image of q with the largest density.

Negative leakage So far we have considered protection against positive leakage only. Negative leakage, which allows one to infer the absence of a tuple (i.e $P(\neg t|O) \gg P(\neg t)$), is allowed under our adversarial privacy definition 2.3. Nonetheless it is possible to extend Algorithm 2.1 to protect against negative leakage as well. This extension guarantees that: $P(\neg t|O) \leq e^\epsilon P(\neg t) + \gamma$. The extension works for a subclass of stable queries, and does so by using a larger λ (the noise parameter). The value of λ has to be $O(\log^g(m))$ rather than $O(\log^{g-1}(m))$ used in Algorithm 2.1.

We explain now the query class for which the extension works. Recall that stable queries are those that have dense derivate queries. The extension works for queries that are both stable and dense. Δ is such a query. However, $\vee$ is not as it is stable, but not dense. In general, if q is a query that counts the number of occurrences of a subgraph h and h is a connected graph with cycles then q is both stable and dense.

5. DISCUSSION & RELATED WORK

In this section we recap our contributions, review related work, and note open questions.

5.1 Recap

We have described an output perturbation mechanism (Algorithm 2.1) that guarantees utility for all input databases and answers a large class of queries much more accurately than competing techniques. In particular, for a number of high sensitivity social network queries, the accuracy bounds improve from polynomial in the number of nodes to polylogarithmic (Table 1). We have shown in the full version of the paper that any ϵ-indistinguishable algorithm that provides utility for all inputs must add noise that is proportional to the global sensitivity. Thus to achieve our utility results, we must relax the privacy standard.

Our relaxed privacy condition is based on *adversarial privacy*, which requires that the posterior distribution be bounded by the prior. For all queries that are *stable* —a class that includes a number of practically important queries—the adversaries we consider believe (with high probability) that the query has low local sensitivity, as formalized by Proposition 3.7. The adversary's belief that the local sensitivity is low, say no more than λ, means that we can return accurate answers—with noise proportional to λ rather than the global sensitivity—and still ensure privacy (Proposition 3.8). Even when the input has high local sensitivity, privacy is preserved because of the novel step of the algorithm that returns a uniformly random answer. The probability of this event is rare, but it is just large enough to thwart the adversary (see details of proof in the full version).

Our first main result (Theorem 2.6) shows that for CQ($\neq$) queries that are *stable*, the algorithm guarantees (ϵ, γ)-adversarial privacy w.r.t. (δ, k)-bounded $\mathcal{PTLM}$ adversaries. Query stability is a syntactic property that we show is related to the expected local sensitivity for this class of adversaries. Our second main result (Thm 2.12) justifies the choice of adversary and makes an important connection between the two notions of privacy: the $\mathcal{PTLM}$ adversaries are precisely the class of adversaries for which any ϵ-indistinguishable mechanism will provide adversarial privacy. This clarifies the relaxation of privacy we consider since our algorithm guarantees privacy only against $\mathcal{PTLM}$ adversaries whose prior distribution is numerically bounded by (δ, k).

We believe the equivalence between adversarial privacy and ϵ-indistinguishability will be of independent interest and could be a fruitful source of new insights. For example, in Sec. 4.1 we use this connection to develop a relaxed version of ϵ-indistinguishability (which is independent of any assumptions about the adversary).

5.2 Related Differential Privacy Results

Ours is not the only approach to accurately answering high sensitivity queries. Nissim et al. [16] recently proposed an output perturbation algorithm that adds instance-based noise. The noise depends on the local sensitivity, but to avoid leaking information about the instance, the noise must follow a smooth upper-bound of the local sensitivity. The hope is that for many database instances (e.g. social networks that occur in practice), a smooth sensitivity bound can be found that is substantially lower than the global sensitivity.

Smooth sensitivity is a different, and complementary, strategy to what we propose here. For queries with high global sensitivity, one cannot guarantee both utility for worst-case inputs *and* privacy against a worst-case adversary (i.e., ϵ-indistinguishability). Nissim et al. attack the problem by relaxing the utility guarantee, motivated by the premise that for typical inputs, local sensitivity is low. We attack the problem by considering weaker (but realistic) adversaries, motivated by the premise that in practice adversaries will not have near-complete knowledge of the input. To date, relatively little is known about the smooth sensitivity of functions of interest. The authors show how to compute the smooth sensitivity for two specific functions (median,

minimum spanning tree) and propose a sampling technique that can be applied to others. Nevertheless, the smooth sensitivity of stable queries such as $h_{i,j}$ may be difficult to compute. In addition, the utility bounds for smooth sensitivity depend on the particular instance and we are not aware of any general analysis that characterizes the set of inputs for which accurate results are possible. Therefore we cannot compare these techniques directly with the results in Table 1.

Ganta et al. [10] define a very different notion of adversarial privacy than we consider here. Roughly, the definition says an algorithm is private if for all adversaries, the adversary's posterior belief does not change much given the change of a single tuple in the database. Privacy is defined as a posterior-to-posterior comparison, rather than prior-to-posterior comparison, although they also show a relationship with differential privacy.

Other Related Work Existing adversarial privacy techniques [6, 17] are designed for tabular data and do not yield accurate query answers for social network data (or for queries with joins more generally).

The private analysis of social networks and other graph data has recently begun to receive attention. The insufficiency of simple anonymization (by removing node names) has been demonstrated: Backstrom et al. [1] proposed a number of attacks on online social networks, and Hay et al. [11] performed a systematic assessment of the threats of structural attacks. A number of anonymization algorithms have been proposed. Each publishes a transformed graph that provides anonymity by creating structurally similar groups [11, 22, 12, 3]. The empirical evaluation of utility looks promising, but no formal guarantees are shown. The primary distinction with the present work is the privacy condition on the output: anonymous data can still leak secret information (e.g., see [13]); whereas our techniques formally bound the information disclosure. Finally, Ying and Wu [20] show that randomly perturbing the graph can provide a notion of adversarial privacy, but empirical results show the utility of the data diminishes rapidly with increased privacy.

5.3 Open Questions and Future Work

Precisely characterizing the relationship between query structure and global sensitivity is an interesting open problem. Our algorithm adds noise less than the global sensitivity for a number of important queries, but is not guaranteed to do so for all queries. An improved algorithm could set λ to the minimum of GS_q and the current quantity. In addition, some queries of interest (e.g. $d_{i,v}$, K_i) have poor utility when i, the number of joins, is large. Acceptable utility for large i currently requires additional restrictions on the adversary, but initial results suggest these restrictions may be avoidable.

6. REFERENCES

[1] L. Backstrom, C. Dwork, and J. M. Kleinberg. Wherefore art thou R3579X?: Anonymized social networks, hidden patterns, and structural steganography. In *WWW*, 2007.

[2] H. W. Block, T. H. Savits, and M. Shaked. Some concepts of negative dependence. In *Ann. of Prob.*, 1982.

[3] A. Campan and T. M. Truta. A clustering approach for data and structural anonymity in social networks. In *PinKDD*, 2008.

[4] C. Dwork. Differential privacy. In *ICALP*, 2006.

[5] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

[6] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS*, 2003.

[7] A. V. Evfimievski, R. Fagin, and D. P. Woodruff. Epistemic privacy. In *PODS*, 2008.

[8] T. Feder and M. Mihail. Balanced matroids. In *STOC*, 1992.

[9] Full version: `http://www.cs.washington.edu/homes/vibhor/relationship_privacy.pdf`.

[10] S. Ganta, S. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *KDD*, 2008.

[11] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. In *VLDB*, 2008.

[12] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *SIGMOD*, 2008.

[13] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *ICDE*, 2006.

[14] G. Miklau and D. Suciu. A formal analysis of information disclosure in data exchange. In *SIGMOD*, 2004.

[15] M. Newman. The structure and function of complex networks. *SIREV: SIAM Review*, 2003.

[16] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, 2007.

[17] V. Rastogi, D. Suciu, and S. Hong. The boundary between privacy and utility in data publishing. In *VLDB*, 2007.

[18] J. G. Shanthikumar and H.-W. Koo. On uniform conditional stochastic order conditioned on planar regions. In *Ann. of Probab.*, 1990.

[19] V. Vu. Concentration of non-lipschitz functions and applications. *RSA*, 2002.

[20] X. Ying and X. Wu. Randomizing social networks: a spectrum preserving approach. In *SIAM*, 2007.

[21] E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationships in graph data. In *PinKDD*, 2007.

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