No Free Lunch in Data Privacy

Daniel Kifer
Penn State University
dan+sigmod11@cse.psu.edu

Ashwin Machanavajjhala
Yahoo! Research
mvnak@yahoo-inc.com

ABSTRACT

Differential privacy is a powerful tool for providing privacy-preserving noisy query answers over statistical databases. It guarantees that the distribution of noisy query answers changes very little with the addition or deletion of any tuple. It is frequently accompanied by popularized claims that it provides privacy without any assumptions about the data and that it protects against attackers who know all but one record. In this paper we critically analyze the privacy protections offered by differential privacy.

First, we use a no-free-lunch theorem, which defines non-privacy as a game, to argue that it is not possible to provide privacy and utility without making assumptions about how the data are generated. Then we explain where assumptions are needed. We argue that privacy of an individual is preserved when it is possible to limit the inference of an attacker about the *participation* of the individual in the data generating process. This is different from limiting the inference about the presence of a tuple (for example, Bob's participation in a social network may cause edges to form between pairs of his friends, so that it affects more than just the tuple labeled as "Bob"). The definition of evidence of participation, in turn, depends on how the data are generated – this is how assumptions enter the picture. We explain these ideas using examples from social network research as well as tabular data for which deterministic statistics have been previously released. In both cases the notion of participation varies, the use of differential privacy can lead to privacy breaches, and differential privacy does not always adequately limit inference about participation.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Statistical Databases; K.4.1 [**Computers and Society**]: Privacy

General Terms

Security

SIGMOD'11, June12–16, 2011, Athens, Greece.

1. INTRODUCTION

Data privacy is an expanding sub-field of data management whose goal is to answer queries over sensitive datasets without compromising the privacy of the individuals whose records are contained in these databases.

One recent breakthrough in this field is a privacy definition called *differential privacy* [10, 8]. Query answering algorithms that satisfy differential privacy must produce noisy query answers such that the distribution of query answers changes very little with the addition, deletion, or modification of any tuple. Recent work has shown that the resulting query answers can enable very accurate privacy-preserving statistical analyses of sensitive datasets [25, 13, 5, 18]. There are two flavors of differential privacy, which we call *unbounded* and *bounded*. The formal definitions are:

DEFINITION 1.1. (Unbounded Differential Privacy [8]). *A randomized algorithm $\mathcal{A}$ satisfies unbounded ϵ-differential privacy if $P(\mathcal{A}(D_1) \in S) \leq e^{\epsilon} P(\mathcal{A}(D_2) \in S)$ for any set S and any pairs of databases D_1, D_2 where D_1 can be obtained from D_2 by either adding or removing one tuple.*

DEFINITION 1.2. (Bounded Differential Privacy [10]). *A randomized algorithm $\mathcal{A}$ satisfies bounded ϵ-differential privacy if $P(\mathcal{A}(D_1) \in S) \leq e^{\epsilon} P(\mathcal{A}(D_2) \in S)$ for any set S and any pairs of databases D_1, D_2 where D_1 can be obtained from D_2 by changing the value of exactly one tuple.*

Bounded differential privacy derives its name from the fact that all of the datasets D_1, D_2 have a fixed size n, while unbounded differential privacy has no such restriction.

Additional *popularized* claims have been made about the privacy guarantees of differential privacy. These include:

- It makes no assumptions about how data are generated.
- It protects an individual's information (even) if an attacker knows about all other individuals in the data.
- It is robust to arbitrary background knowledge.

These are loose, though popular, interpretations of formal guarantees provided by differential privacy. In this paper, we critically analyze the guarantees of differential privacy.

We start with a no-free-lunch theorem, which defines non-privacy as a game, and which states that it is impossible to provide privacy and utility without making assumptions about the data. This is a reformulation, simplification, and re-interpretation of an impossibility result of Dwork and Naor [8, 11] (by removing dependence on cryptographic techniques and related complications, the result is accessible to

a wider audience). This shows that assumptions about the data are needed for differential privacy to avoid this game-theoretic notion of non-privacy. For comparison, we present a variation of differential privacy which does guarantees privacy without assumptions but which provides very little utility.

Then we argue that a major criterion for a privacy definition is the following: can it hide the evidence of an individual's participation in the data generating process? That is, can it limit the ability of an attacker to infer that an individual has participated? The notion of *evidence of participation* depends on how the data are generated and is different from the presence or absence of a tuple - for example, Bob's participation in a social network can cause links to form between pairs of Bob's friends and so the participation affects more than just the tuple marked "Bob". Formalizing the notion of participation is extremely difficult, but as with non-privacy, it is easier to identify cases where evidence of participation is not protected from inference. We do this with social networks (Section 3) and tabular data for which deterministic statistics have been previously released (Section 4). In each case, the notion of participation is different: in social networks, an individual can influence other people's data; for tabular data, the released query answers already provide some evidence of participation. We believe that under any reasonable formalization of *evidence of participation*, such evidence can be encapsulated by exactly one tuple only when all tuples are independent (but not necessarily generated from the same distribution). We believe this independence assumption is a good rule of thumb when considering the applicability of differential privacy.

Relying on the principle that the privacy of an individual is preserved whenever we can prevent inference about the individual's participation, we see that the main guarantee of differential privacy (i.e. the distribution of noisy query answers changes very little with the addition, deletion, or modification of a tuple) does not always guarantee privacy since deletion of a tuple is not the same as removal of evidence of participation. Since evidence of participation requires additional assumptions about the data (as we demonstrate in detail in Sections 3 and 4), this addresses the first popularized claim – that differential privacy requires no assumptions about the data.

The second popularized claim is better interpreted as stating that differential privacy limits (up to a multiplicative factor) the probabilistic inference of an attacker who knows all but one record in a dataset. This claim implicitly assumes that the more an attacker knows, the greater the privacy risks. We show that this latter assumption is flawed by (1) creating variations of differential privacy where the corresponding attackers have more knowledge yet less noise is added to query answers and (2) showing that this leaks more private information to attackers with less prior knowledge. Thus when choosing privacy definitions based on resistance to certain classes of attackers, the right choice of attackers is crucial for meaningful privacy guarantees, but the right choice is by no means obvious. We argue that the attackers to consider are those from whom it is possible to hide (i.e., limit inference about) the participation of an individual in the data generating process. This is different from limiting inference about the presence or absence of a tuple t in a dataset, especially when other tuples are correlated with t; those other tuples may provide evidence about the participation of t and this evidence must be hidden as well. With this interpretation, we should not simply assume that an attacker already has this evidence since more knowledgeable attackers are not always the biggest threats.

The third claim, that differential privacy is robust to arbitrary background knowledge, has been formalized and studied by [19]. More accurately, differential privacy is robust when certain subsets of the tuples are known by an attacker. However, many reasonable types of background knowledge can cause privacy breaches when combined with differential privacy (in other words, differential privacy composes well with itself [14] but not necessarily with other privacy definitions or data release mechanisms). Such background knowledge includes previously released exact query answers. We demonstrate a privacy breach using this background knowledge in Section 4 and we propose solutions that account for those previously released query answers.

The goal of this paper is to clear up misconceptions about differential privacy and to provide guidelines for determining its applicability. In particular our contributions are that we (1) simplify a result of Dwork and Naor [8, 11] in order to show how privacy relies on assumptions about data generation, (2) propose a participation-based guideline – "does deleting an individual's tuple erase all evidence of that individual's participation in the data-generating process?" – for determining whether differential privacy is suitable for a given application, (3) demonstrate that differential privacy does not meet this guideline when applied to arbitrary social networks and show how this can result in a privacy breach, (4) demonstrate that differential privacy does not meet this guideline when applied to tabular data when an attacker has aggregate-level background knowledge (and show a potential privacy breach), and (5) propose a modification of differential privacy that avoids privacy breaches for tabular data with aggregate-level background knowledge.

The outline of this paper is as follows. We discuss the no-free-lunch theorem and the game-theoretic definition of non-privacy in Section 2, where we also show that more knowledgeable attackers do not necessarily present the greatest risks to privacy. We specialize this discussion to social networks in Section 3, where experiments with several popular social network models show that differential privacy does not adequately limit inference about the participation of an edge in the network. We then consider tabular data, with a side release of global statistics, in Section 4. The notion of participation is very different here because some evidence of participation is already provided by the released statistics. We propose a generalization differential privacy to take into account those statistics to limit further privacy breaches, and algorithms that satisfy our generalizations.

2. ANALYSIS OF ATTACKERS

2.1 The No-Free-Lunch Theorem

In this section we argue that it is not possible to guarantee privacy and utility without making assumptions about the data-generating mechanism.

2.1.1 Utility

To capture the notion of utility, we define a *lower bound* on utility called the *discriminant*. It measures whether there exists *any* query that can be answered reasonably accurately.

DEFINITION 2.1. (Discriminant ω). *Given an integer $k > 1$, a privacy-infusing query processor $\mathcal{A}$, and some constant c, we say that the discriminant $\omega(k, \mathcal{A}) \geq c$ if there exist k possible database instances $D_1, \ldots, D_k$ and disjoint sets $S_1, \ldots, S_k$ such that $P(\mathcal{A}(D_i) \in S_i) \geq c$ for $i = 1, \ldots, k$ (the randomness only depends on $\mathcal{A}$).*

We illustrate this definition with an example. Suppose we are asking a query about how many cancer patients are in the data. If the privacy-infusing query processor $\mathcal{A}$ can answer this query with reasonable accuracy, then the discriminant $\omega(k, \mathcal{A})$ is close to 1. To see this, we set $k = 3$ (for example) and D_1 to be any database with 0 cancer patients, D_2 to be any database with $10,000$ cancer patients and D_3 to be any database with $20,000$ cancer patients. Suppose $\mathcal{A}$ works as follows: if there are 0 cancer patients, it outputs some number in the range [0, 1000] with probability 0.95; if there are $10,000$ cancer patients, it outputs some number in the range [9000, 11000] with probability 0.95; if there are $20,000$ cancer patients it outputs a number in the range [19000, ∞] with probability 0.95. Then we set $S_1 = [0, 1000]$ (in Definition 2.1), $S_2 = [9000, 11000]$ and $S_3 = [19000, \infty]$. Since $P(\mathcal{A}(D_i) \in S_i) \geq 0.95$ and the S_i are pairwise disjoint we must have $\omega(k, \mathcal{A}) \geq 0.95$.

From this example we see that if $\omega(k, \mathcal{A})$ is close to one, then we may get a reasonably useful answer to some query (or we may not, this is only a lower bound). On the other hand, if $\omega(k, \mathcal{A})$ is close to $1/k$ (which can happen when the output of $\mathcal{A}$ is uniformly distributed and does not depend on the data), then we *cannot* get useful answers to *any* query.

Since the goal of modern privacy definitions is to answer queries relatively accurately, they all allow the use of algorithms with discriminants close to 1. For example, the discriminant of proposed k-anonymity algorithms [31] is 1.

Proposed differentially private algorithms also have discriminants close to or equal to 1. For example, the Laplace Mechanism [8] is an algorithm satisfying differential privacy that answers a count query by adding noise from the Laplace distribution with density function $\frac{\epsilon}{2}e^{-|x|\epsilon}$.

PROPOSITION 2.1. *The discriminant $\omega(k, \mathcal{A})$ of the Laplace Mechanism for unbounded differential privacy is 1 for any k. For bounded differential privacy, the discriminant of the Laplace Mechanism becomes arbitrarily close to 1 as the data size increases.*

PROOF. Fix k and $\delta > 0$ and let X be a random variable with density function $\frac{\epsilon}{2}e^{-|x|\epsilon}$. Choose n large enough so that $P(X > 2n/3k) < \delta$. For $i = 1, \ldots, k$ let D_i be a database with in/k cancer patients and define $S_i = [in/k - n/3k,\ in/k + n/3k]$ (an interval centered around in/k with width $2n/3k$). Note that all the S_i are disjoint. We will use the Laplace Mechanism $\mathcal{A}$ to answer the query "how many cancer patients are there". By construction, for any r, $P(\mathcal{A}(D_i) \in S_i) > 1 - \delta$. Thus $\omega(k, \mathcal{A}) \geq 1 - \delta$. Since δ is arbitrary, we get $\omega(k, \mathcal{A}) = 1$ for unbounded differential privacy. The result for bounded differential privacy follows since we can make δ arbitrarily small by making n large. □

2.1.2 Non-privacy

Motivated by Dwork and Naor's proof [8, 11] on the impossibility of Dalenius' vision of statistical privacy [6], we define a game between an attacker and a data curator. Let $\mathcal{D}$ be a database schema and let q be a *sensitive query*, with k possible outputs, such that $q(D)$ is considered sensitive for any database instance D. For example, $q(D)$ may return the first record in D. Let $\mathcal{P}$ be a data-generating mechanism. The game proceeds as follows.

DEFINITION 2.2. (Non-Privacy Game). *The data curator obtains a database instance D from the data-generating mechanism $\mathcal{P}$. The data curator provides the attacker with the output $\mathcal{A}(D)$ of a privacy-infusing query processor $\mathcal{A}$. The attacker then guesses the value of $q(D)$. If the guess is correct, the attacker wins and non-privacy is achieved.*

The following no-free-lunch theorem states that if there are no restrictions on the data-generating mechanism and if the privacy-infusing query processor $\mathcal{A}$ has sufficient utility (discriminant close to 1) then there exists a data-generating mechanism $\mathcal{P}$ such that the attacker's guess, without access to $\mathcal{A}(D)$, is correct with probability $1/k$ (k is the number of possible outputs of the query).[1] However, with access to $\mathcal{A}(D)$, the attacker wins with probability close to 1.

THEOREM 2.1 (NO FREE LUNCH). *Let q be a sensitive query with k possible outcomes. Let $\mathcal{A}$ be a privacy-infusing query processor with discriminant $\omega(k, \mathcal{A}) > 1 - \epsilon$ (for some $\epsilon > 0$). Then there exists a probability distribution $\mathcal{P}$ over database instances D such that $q(D)$ is uniformly distributed but the attacker wins with probability at least $1-\epsilon$ when given $\mathcal{A}(D)$.*

PROOF. Choose database instances $D_1, \ldots, D_k$ such that $\forall i, q(D_i) = i$. Choose sets $S_1, \ldots, S_k$ as in Definition 2.1 (discriminant) so that $\omega(k, \mathcal{A}) \geq \min_i P(\mathcal{A}(D_i) \in S_i) > 1-\epsilon$, with the probability depending only on the randomness in $\mathcal{A}$. Let $\mathcal{P}$ be the uniform distribution over $D_1, \ldots, D_k$. The attacker's strategy is to guess $q(D) = i$ if $\mathcal{A}(D) \in S_i$ and to guess randomly if $\mathcal{A}(D)$ is not in any of the S_i. Clearly the attacker wins with probability at least $1 - \epsilon$. □

This theorem can be viewed as a weaker reformulation and significant simplification of Dwork and Naor's result [8, 11]: we eliminated the use of auxiliary side information, introduced a more natural notion of utility, and gave a shorter and transparent proof. It is now clear why assumptions are needed: without restrictions on $\mathcal{P}$, we cannot guarantee that the privacy-infusing query processor $\mathcal{A}$ avoids non-privacy.

2.1.3 No-Free-Lunch and Differential Privacy

We now explain how the no-free-lunch theorem relates to differential privacy and evidence of participation and how an assumption that $\mathcal{P}$ generates records independently can prevent an attacker from winning against a differentially private algorithm. Suppose Bob's sensitive attribute R can take one of the values $1, \ldots, k$. Consider a $\mathcal{P}$ which generates data where Bob's record always appears and when Bob's sensitive attribute $R = j$ then there are $j \times 10,000$ cancer patients in the data. An attacker can ask "how many cancer patients are there?" and to satisfy 0.1-differential privacy, the query processor $\mathcal{A}$ can add Laplace(10) noise to the true answer (with variance 200). The attacker then can divide the noisy answer by $10,000$, round to the nearest integer j, and, with high probability, this will be the correct value of Bob's sensitive attribute R (since $200 << 10,000$). In this case,

[1] The theorem easily extends to probabilities other than $1/k$.

the number of cancer patients provide evidence about Bob's participation.

This example relies on a correlation between Bob's sensitive attribute and the other records in the database to provide evidence of participation. In this particular case, such a correlation seems artificial and so we are likely to *assume* that there is no such dependence between records, so that the number of cancer patients can no longer provide evidence about Bob. If we assume that $\mathcal{P}$ generates records independently (and if the assumption is correct), the attacker would not be able to guess Bob's sensitive attribute from the number of cancer patients. Thus under the assumption of independence of records, differential privacy would be safe from this attack. In Sections 3 and 4 we will see examples of correlations which are not artificial and for which the use of differential privacy can lead to privacy breaches.

Before concluding this section, note that there do exist privacy definitions which make no assumptions about the data. One example is:

DEFINITION 2.3. (Free-Lunch Privacy). *A randomized algorithm $\mathcal{A}$ satisfies ϵ-free-lunch privacy if for* any *pair of database instances D_1, D_2 (not just those differing in one tuple) and for any set S, $P(\mathcal{A}(D_1) \in S) \leq e^{\epsilon} P(\mathcal{A}(D_2) \in S)$.*

It is not hard to see that the discriminant $\omega(k, \mathcal{A})$ of any algorithm $\mathcal{A}$ satisfying ϵ-free-lunch privacy is bounded by $\frac{e^{\epsilon}}{k-1+e^{\epsilon}}$. Notice that this is bounded away from 1 whether we restrict the size of the database or not. In contrast, virtually all noise-infusing privacy definitions (such as bounded differential privacy, randomized response [32], etc.) have discriminants that become arbitrarily close to 1 as the data size increases. Thus with free-lunch privacy, we would have difficulty distinguishing between small and large databases (let alone finer details such as answers to range queries). This lack of utility is also clear from its definition; this is the price we must pay for making no assumptions about the data-generating mechanism.

2.2 Knowledge vs. Privacy Risk

Before discussing privacy breaches in social networks and tabular data (with pre-released exact query answers) in Sections 3 and 4, we present in this section a general guideline for determining whether a privacy definition is suitable for a given application. This guideline is based on "hiding evidence of participation" of an individual from an attacker. In many cases this guideline calls for limiting the inferences made by attackers that are *less* knowledgeable than those considered by differential privacy (i.e. attackers who know about all but one tuple in a database). Somewhat counterintuitively, this can actually *decrease* the probability of disclosing sensitive information. Thus we first show that privacy definitions that limit the inference of more knowledgeable attackers can sometimes leak more sensitive information than privacy definitions that protect against less knowledgeable attackers. Then we present our "evidence of participation" guideline.

Let us start with two more forms of differential privacy – attribute and bit differential privacy.

DEFINITION 2.4. (Attribute Differential Privacy). *An algorithm $\mathcal{A}$ satisfies ϵ-attribute differential privacy if for every pair D_1, D_2 of databases such that D_2 is obtained by changing one attribute value of one tuple in D_1, we must have $P(\mathcal{A}(D_1) \in S) \leq e^{\epsilon} P(\mathcal{A}(D_2) \in S)$ (for any set S).*

DEFINITION 2.5. (Bit Differential Privacy). *An algorithm $\mathcal{A}$ satisfies ϵ-attribute differential privacy if for every pair D_1, D_2 of databases such that D_2 is obtained by changing one bit of one attribute value of one tuple in D_1, we must have $P(\mathcal{A}(D_1) \in S) \leq e^{\epsilon} P(\mathcal{A}(D_2) \in S)$ (for any set S).*

Attribute differential privacy corresponds to an attacker who has full information about the database except for one attribute of one record. Bit differential privacy corresponds to an attacker who knows everything except one bit in the database encoding. Both definitions limit the attacker's inference about the remainder of the data. Clearly the attacker for bit-differential privacy knows more than the attacker for attribute-differential privacy who knows more than the corresponding attacker for bounded-differential privacy (Definition 1.2) who only knows all records but one.

It is easy to see that an algorithm $\mathcal{A}$ that satisfies bounded-differential privacy also satisfies attribute-differential privacy and an algorithm that satisfies attribute-differential privacy also satisfies bit-differential privacy. In these cases, an algorithm that protects against (i.e. limits the inference of) a less knowledgeable attacker also protects against the more knowledgeable attacker. However, we now show that an algorithm that limits the inference of a more knowledgeable attacker can leak private information to a less knowledgeable attacker.

Consider the following table with 1 tuple (Bob) and two 2-bit attributes R_1 and R_2:

	$\mathbf{R_1}$	$\mathbf{R_2}$
Bob	00	10

There are 16 possible databases instances with 1 tuple. Bounded-differential privacy allows an algorithm $\mathcal{A}_1$ to output Bob's tuple with probability $\frac{e^{\epsilon}}{15+e^{\epsilon}}$ and any other tuple with probability $\frac{1}{15+e^{\epsilon}}$. Attribute-differential privacy allows an algorithm $\mathcal{A}_2$ to output Bob's record with probability $\frac{e^{2\epsilon}}{9+6e^{\epsilon}+e^{2\epsilon}}$ (each of the 6 records where only one attribute differs from Bob can be output with probability $\frac{e^{\epsilon}}{9+6e^{\epsilon}+e^{2\epsilon}}$, and each of the 9 remaining records can be output with probability $\frac{1}{9+6e^{\epsilon}+e^{2\epsilon}}$). The allowable distribution for bit-differential privacy is more complicated, but Bob's record can be output with probability $\frac{e^{4\epsilon}}{e^{4\epsilon}+4e^{3\epsilon}+6e^{2\epsilon}+4e^{\epsilon}+1}$. Figure 1 plots the different probabilities of outputting Bob's record as ϵ ranges from 0 to 1. We clearly see that bit-differential privacy, with which we associate the most knowledgeable attacker, outputs Bob's record with the highest probability, followed by attribute-differential privacy and followed by bounded-differential privacy (which we associate with the least knowledgeable attacker).

The intuition about why bit-differential privacy leaked the most amount of private information is that, since the attacker already knows so much about Bob's record, there is little need to output a record drastically different from it. There is very little left for the attacker to learn and so we can concentrate *most* of the randomness on this area of the attacker's uncertainty, which would be the set of records that differ from Bob's by 1 bit. Another example is an attacker who knows everything: we can release the entire database without the attacker learning anything new.

We can extend these ideas to correlated data. Consider the following example:

EXAMPLE 2.1. *Bob or one of his 9 immediate family members may have contracted a highly contagious disease, in which case the entire family would have been infected. An*

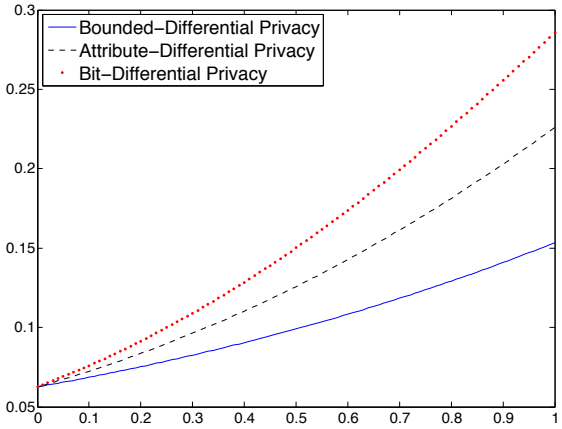

Figure 1: Probability (y-axis) of outputting Bob's record as ϵ (x-axis) varies for bit-differential privacy (top line), attribute-differential privacy (middle line), bounded-differential privacy (bottom line)

attacker who knows all 9 family members have this disease already has strong evidence that Bob is infected. A weaker attacker who knows nothing about the family's health can ask the query "how many in Bob's family have this disease?". The true answer is almost certainly going to be either 0 *or* 10. *Suppose Laplace*$(1/\epsilon)$ *noise is added to the true answer and that the resulting differentially private query answer was* 12. *The answer* 12 *is* $e^{10\epsilon}$ *times more likely when the true answer is* 10 *(and hence Bob is sick) than when the true answer is* 0 *(and Bob is healthy). Thus data correlation and a differentially private answer produced a result where a weaker attacker's probability estimate (of Bob being sick) can change by a (large) factor of* $e^{10\epsilon}$. *In contrast, differential privacy guarantees that the stronger attacker (who already knows 9 family members are sick) can change his probability estimate by a factor of at most* e^{ϵ}.

Other types of correlations are also possible. In Section 3 we show how correlations between edges in a social network can lead to a privacy breach, and in Section 4 we show that prior release of deterministic aggregate statistics can cause tuples to become correlated and that this can also lead to a privacy breach.

2.2.1 Participation-based guidelines

The key feature of Example 2.1 was that Bob's tuple had influence over other tuples in the data. Deleting Bob's tuple would not remove Bob's *influence* on the data and so the rest of the data provides *evidence* about Bob and his health. This evidence comes from Bob's participation in the data generating mechanism: Bob's interaction with family members (i.e. participation) and the mechanisms governing infection and growth of pathogens (i.e. other relevant aspects of data generation).

Thus if privacy guarantees are based on limiting the inference of a class of attackers, this class must be chosen wisely. We argue that an important consideration when evaluating a privacy definition is the following question: how well can it hide the *evidence of participation* of an individual from an attacker who has little prior evidence of this individual's participation? When data records are independent, then deleting one record eliminates all evidence of its participation. In such as case, differential privacy is appropriate since it guarantees that differentially private query answers are barely affected by record deletion. On the other hand, when records are not assumed to be independent, record deletion may not hide evidence of participation (in which case differential privacy may not be suitable). In general, it is the data-generating mechanism which determines what properties of the data provide evidence of participation (as we show in Sections 3 and 4). Thus the definition of participation and hence the suitability of various privacy definitions actually depends on how the data are generated.

It is difficult to create a general definition of "participation" that applies to all data-generating mechanisms. One of the difficulties is that if aggregate statistics about the data have already been released, then those statistics already provide *some* evidence of Bob's participation; protecting the *rest* of this evidence seems to entail some concept of *degree* of participation. Thus instead of a general definition, we consider special cases. In Section 3, where we show that differential privacy does not always hide the participation of an *edge* in a social network, we show that the presence or absence of one edge can have a dramatic effect on the subsequent growth of a network. In Section 4, where we discuss prior release of statistics, we show that a number of other records need to be modified in order to alter evidence about an individual without an attacker realizing it (thus providing a notion of plausible deniability).

3. SOCIAL NETWORKS

In this section we investigate how well differential privacy can hide (limit inference about) the participation of entities in social networks. There are two natural choices for such entities: nodes and edges. A standard application of differential privacy would try to prevent inference about whether or not a node (or edge) has been subsequently removed from the data. For nodes, a differentially private algorithm $\mathcal{A}_{node}$ must satisfy $P(\mathcal{A}_{node}(D_1) \in S) \leq e^{\epsilon} P(\mathcal{A}_{node}(D_2) \in S)$ for all sets S and for all pairs of databases D_1 and D_2 such that one can be constructed from the other by removing a node and all of its incident edges. For edges, a differentially private algorithm $\mathcal{A}_{edge}$ must satisfy $P(\mathcal{A}_{edge}(D_1) \in S) \leq e^{\epsilon} P(\mathcal{A}_{edge}(D_2) \in S)$ for all sets S and for all pairs of databases D_1 and D_2 such that one can be constructed from the other by removing an edge.

It is widely believed that differentially private algorithms $\mathcal{A}_{node}$ that try to limit inference about nodes can only return query answers that are too noisy for practical applications [17], and so it is common to sacrifice privacy for more utility by using differentially private algorithms $\mathcal{A}_{edge}$ for edges [12]. Thus in this section we will examine whether limiting inference about whether an edge has been subsequently removed is the same as limiting inference about the participation of that edge in the social network.

It is difficult to formulate a general and formal definition for what *evidence of participation* means, but it is very easy to identify special cases where this evidence is clearly not protected from inference. One such example is the following. Suppose a social network starts with two distinct communities that do not interact (they share at most one edge); one reason is that they may be unaware of each other. The attacker knows this, and also knows that if there is an edge between the two communities, then it is between Bob,

a member of Community 1, and someone else in Community 2. Let us define a privacy breach to be the event that the attacker can determine (with high probability) whether or not Bob had an edge to someone in the second community. We will let these communities grow, using three different social network models [22, 21, 24], after which the attacker will ask the query "how many edges are there between the two communities". We now consider (1) how deleting Bob's edge from the database affects the query answer and (2) how the answer would have been affected if Bob's had not existed. Deleting Bob's edge simply changes the answer by 1. Differential privacy would guarantee that if the true answer were either X or $X-1$, the noisy answer would be the same, with high probability, in either case. Now let us consider the effect of the participation of Bob's edge in the network. Suppose that Bob's edge was incident to Charlie (from Community 2). Because of their friendship, Bob introduces Alice (from Community 1) to Charlie. Alice then introduces her friends (from Community 1) to Charlie, and so on. Thus Bob's initial edge could have caused many additional edges to form (and most of them would not even be incident to Bob). If Bob did not have that initial edge, it is possible that those additional edges would not have formed and so the query answer could have been significantly different. To check how different, we use several popular social network models [22, 21, 24] to simulate this "what if" scenario.

For each model and parameter setting, we ran the model starting from a network where there is an edge between Bob and a member of a different community and also in the case where no such edge exists. In each case we run the model $1,000,000$ times to get an accurate estimate of the expected number of edges between the two communities (with and without that initial edge) after the network has being growing for a while. The difference Y, between the expected number of edges when Bob had a link to another community and when he did not, is a measure of the degradation of the differential privacy guarantees. It is also a measure of the causal influence of Bob's edge. A naïve application of ϵ-differential privacy would add Laplace$(1/\epsilon)$ noise (with variance $2/\epsilon^2$) to the query "how many edges are there between the two communities". However, to prevent inference about the participation of Bob's edge, at least Laplace(Y/ϵ) noise (with variance $2Y^2/\epsilon^2$) would have to be added (more noise could be needed, since Y is only the difference in expectation, and there could be more revealing queries). Furthermore, we would not know Y a priori unless we knew the model parameters (i.e. made assumptions about how the data were generated). Otherwise, we would have no guidance on how much noise to add.

Forest Fire Model [22]. The first model we use is the forest fire model [22]. We modify it slightly to make it more fair to differential privacy (in the original version, if there was no edge between two communities, no such edge would ever form). When using this model, our initial network has 10 nodes (5 per community) with three random within-community outlinks per node. We grow this network until it has 50 nodes as follows. When a new node enters the network, it picks an ambassador node uniformly at random. The new node then joins the same community as the ambassador and forms an edge to the ambassador. The new node picks a number k from a geometric distribution with mean $p/(1-p)$ and forms an edge with k of the ambassador's outlinks. The new node uses the same process to form an edge with some of the ambassador's inlinks. Now the new node repeats the link generation process with its new neighbors (duplicate edges between nodes are not allowed). Finally, with probability q (which we set to 0.1) the new node adds a link to a randomly chosen node (this modification allows for new edges to form between communities that had no edges before). We used the parameters investigated by Leskovec et al. [22] where the parameter for the geometric distribution for outlinks is 0.37 and for inlinks it is 0.32. When Bob had no edge to another community, the expected number of cross-community edges (after the network grew to a size of 50) is ≈ 119. However, when Bob did initially have an edge to another community, the expected number of cross-community edges was ≈ 240. This is a significant difference, and ϵ-differential privacy would need an extremely small ϵ value to hide the participation of Bob's initial edge (thereby adding a large amount of noise); ϵ should be at least $240-119=121$ times smaller than a naïve application of differential privacy would suggest. In addition, this difference in number of cross-community edges depends on the model parameters, and so we cannot set ϵ reliably *unless we know the parameters of the network-generating distribution.* Even for the particular parameters we chose, considering the small size of the network (50 nodes), such a value of ϵ would add large amounts of noise to any query, thus providing virtually no information about the social network. Thus, in order to use differential privacy to provide meaningful information about social networks, we would have to assume that the network was *not* generated by a forest fire model. Alternatively, a naïve application of differential privacy which did not take into account possible large effects of causality would let our attacker infer the participation of Bob's edge.

Copying Model [21]. The second model we use is the copying model [21]. This model does not grow as fast as the forest fire model because the out-degree of each node is bounded. The network again starts with 10 nodes (5 per community) and set we the out-degree of each node to be 3. The network grows as follows. When a new node enters the network, it chooses a prototype node uniformly at random and joins the same community as the prototype. For each outlink neighbor of the prototype, the new node links to that neighbor with probability $1-p$ and links to a random node with probability p. We ran two sets of experiments. In the first set, we fixed the final network size to be 50 and varied the random link probability p. The results are shown in Figure 2. In the second set, we fixed the random link probability p to 0.05 and varied the final network size. The results are shown in Figure 3.

When the random link probability increases (Figure 2), the copying model begins to degenerate into a random graph model where edges are chosen randomly and so the effect of causality decreases. However, as the network size grows for a fixed p, so does the difference in the expected value of cross-community edges (Figure 3). The growth is modest compared to the forest fire model, but makes it difficult for differential privacy to limit inference about Bob's initial edge. The reason is that the effect of causality grows with time and so the noise parameter ϵ would depend on the size of the final network (and other parameters). Again, it is not possible to set ϵ reliably unless we know or make assumptions about the actual parameters of the data generating

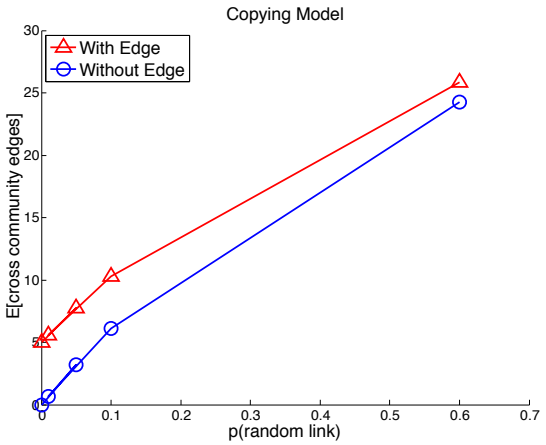

Figure 2: Causal effects in the Copying Model as a function of the random link probability.

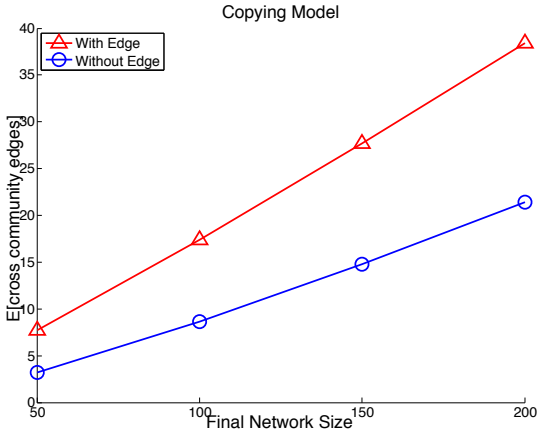

Figure 3: Causal effects in the Copying Model as a function of the final network size

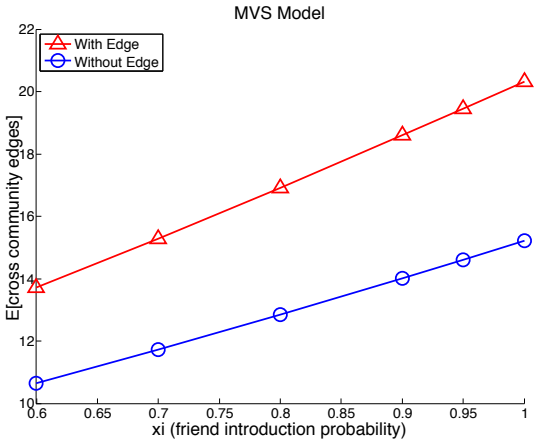

Figure 4: Causal effects in the MVS Model as a function of ξ, the friend-introduction probability.

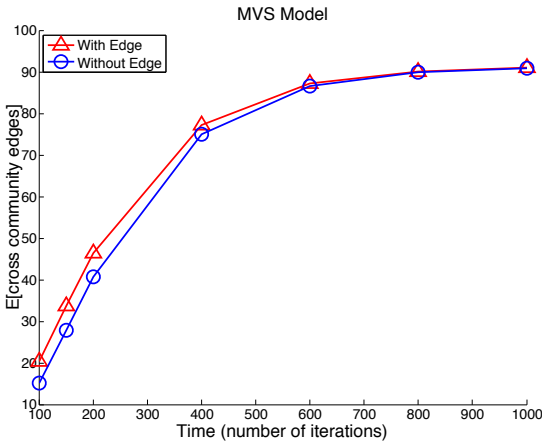

Figure 5: Causal effects in the MVS Model as a function of time (nearness to steady state).

distribution. The alternatives are to set ϵ very small (thus destroying utility), or to ignore causality (i.e. naïve apply differential privacy) and possibly reveal the participation of Bob's edge.

MVS Model [24]. Our final model is the MVS model [24]. In contrast to the forest fire and copying models, this model is actually an ergodic Markov chain (whose state space consists of graphs) and has a steady state distribution. Thus the number of edges does not have a tendency to increase over time as with the other two models. The original MVS model is a continuous time process, which we discretize in a way similar to [24]. In this model, the number of nodes remains constant. At each timestep, with probability η, a link is added between two random nodes. With probability ξ, a randomly selected node selects one of its neighbors at random and asks for an introduction to one of its neighbors. Finally k edges are deleted, where k is a random variable drawn from a Poisson(λ) distribution. Initially both communities have an equal number of nodes and community membership does not change. Initially each node has 2 within community links. We perform two sets of experiments. In the first set of experiments, the network has 20 nodes and we vary ξ, the friend-introduction probability. We compute the number of cross-community edges after 100 time steps (before the network has reached a steady state). This is shown in Figure 4. In the second set of experiments, we set the network size to be 20, we set $\xi = 1$, and we vary the number of iterations to let the network achieve its steady state. This is shown in Figure 5. Note that once the process achieves steady state, the initial conditions (or, more generally, network configurations in the distant past) are irrelevant and the evidence that other edges provide about the participation of Bob's initial link is automatically destroyed by the network model itself. However, there can still be evidence about edges formed in the not-so-distant past.

We see from Figure 4, that depending on network parameters, the expected difference in cross-community edges can be moderately large, but we see that this difference disappears as the network is allowed to reach its steady state (Figure 5). Thus differential privacy may be a reasonable choice for limiting inference about the participation of Bob's edge in such a scenario. However, applying differential privacy here still requires the assumption that the data were gener-

ated by such a favorable model (as opposed to a forest fire model, for example).

4. CONTINGENCY TABLES

In this section we consider privacy-preserving query answering over a table for which some deterministic statistics have already been released. In addition to motivating this problem, we show that applications of differential privacy in this setting can lead to a privacy breach beyond the possible breach caused by the initial release of those statistics. We then propose a solution by modifying differential privacy to take into account such prior deterministic data releases.

The U.S. Census Bureau is one of the largest consumers of privacy technology. It both collects and disseminates data about the U.S. population. Privacy is an important issue so many of the datasets released to the public have been perturbed, masked, or otherwise modified to protect the confidentiality of individuals in the data.

In some cases, utility is more important than privacy. One example is in the release of population counts which are used for apportionment (allocating seats in the House of Representatives to the states). Since this is a highly-charged political issue with significant consequences, these counts must be as accurate as possible so that sampling (which can be broadly interpreted to include noise infusion and complicated statistical data editing) is prohibited. Of particular significance is the 2002 Supreme Court case Utah v. Evans (our short discussion here is based on [3]). This case concerned the year 2000 Decennial Census and the way in which it dealt with missing and inconsistent data. The Census Bureau used statistical imputation to adjust counts for households for which data were not reported. This adjustment was enough to add one seat in the House of Representatives to North Carolina at the expense of Utah. In the Supreme Court case, the State of Utah unsuccessfully argued that statistical imputation should have been considered a form of sampling and therefore disallowed. Even though Utah lost this case illustrates the contentiousness of even a small perturbation of the data.

This case illustrates an interesting application for privacy technology. In some cases utility is so important that some *exact* statistics about the data must be released. After this initial data release, additional queries can be answered, with degraded accuracy, as long as they do not introduce an additional privacy breach (note that this is a special case of the dual privacy problem where the goal is to provide as much privacy subject to utility constraints [15]). For example:

EXAMPLE 4.1. *Due to overriding utility concerns, a university releases the number of male and female employees, a histogram of employee salaries, and a separate histogram for each additional attribute in its employees table. An economist may later seek to determine if salary and gender are independent of each other at this institution. This question cannot be answered from the released statistics but could be answered from the original data by using the chi-squared test (for example). Thus the university would like to provide a privacy-preserving estimate of the chi-squared value on the original data. The release of this approximate chi-squared value should not lead to additional leakage of privacy (when combined with the previously released histograms).*

Differential privacy does not allow for any deterministic release of statistics, so the initial data release in Example 4.1 falls outside of this framework. However, differential privacy might later be used to release a perturbed version of the chi-squared statistic. In cases like these, privacy guarantees may degrade. We provide a simple demonstration in Section 4.1.

4.1 A Privacy Leak

We now show that when deterministic statistics have been previously released, applying differential privacy to subsequent data releases is not enough to prevent *additional* privacy breaches (beyond the possible breaches caused by the initial release of statistics). The reason is that those statistics could add correlations into the data and into future queries that otherwise would be unrelated.

Consider a table T with one attribute R that can take values $r_1, \ldots, r_k$. Suppose that due to overriding utility concerns, the following $k-1$ queries have been answered exactly:

```
SELECT COUNT(*) FROM T WHERE R=r_1 OR R=r_2
                    ⋮
SELECT COUNT(*) FROM T WHERE R=r_{j-1} OR R=r_j
                    ⋮
SELECT COUNT(*) FROM T WHERE R=r_{k-1} OR R=r_k
```

This does not provide enough information to always reconstruct the original table (there are k unknowns but $k-1$ linear equations). However, if we knew the answer to "SELECT COUNT(*) FROM T WHERE R=r_i" for any r_i, then we would be able to exactly reconstruct the table T. In this way the tuples are correlated and we are one exact query answer away from a complete privacy breach (i.e. reconstruction of the table).

Now, differential privacy would allow us to answer the following k queries by adding independent Laplace($1/\epsilon$) noise (with variance $2/\epsilon^2$) to each query [8]:

```
SELECT COUNT(*)+noise FROM T WHERE R=r_1
                    ⋮
SELECT COUNT(*)+noise FROM T WHERE R=r_j
                    ⋮
SELECT COUNT(*)+noise FROM T WHERE R=r_k
```

By themselves, without use of the previously released exact queries, these noisy query answers are innocuous and have little dependence on the value of Bob's tuple. In fact, the only query that would have even a slight dependence on Bob is the query where r_i matched Bob's attribute value.

However, because of the previous release of exact query answers, *each* noisy query now leaks some small amount of information about *every tuple*. To see this, note that simple linear algebra and a noisy answer to "number of people with $R = r_i$", with noise variance $2/\epsilon^2$, allows us to compute a noisy estimate for "number of people with $R = r_1$" also with variance $2/\epsilon^2$. Thus for each of these k noisy queries we can estimate the number of tuples with $R = r_1$ (with variance $2/\epsilon^2$). By averaging over all of these estimates, we get a better estimate for the number of tuples with $R = r_1$ with variance $2/(k\epsilon^2)$, and in fact we can estimate of the number of tuples with $R = r_i$ (for any i) with variance $2/(k\epsilon^2)$.

When k is large (for example R is a d-bit attribute with 2^d possible values), the variance is small so that the table T is reconstructed with very high probability, thus causing a complete privacy breach. This was only possible because

additional noisy queries were answered without taking into account prior release of information.

Thus the guarantee that the distribution of future noisy query answers changes very little if one deletes or arbitrarily modifies one tuple is not sufficient to prevent a privacy breach when exact query answers had been previously released – those exact answers add correlations between the noisy answers that otherwise might not exist.

4.2 A Plausible Deniability

The result of Section 4.1 shows that once deterministic query answers have been released released, Bob (if his information is in the data) can no longer opt out of the data or opt to change his tuple arbitrarily – opting out would have only been meaningful *before* any query answers were released – so the differential privacy guarantees for future query answers would not be as compelling to Bob. Bob can still try to limit *further* damage by opting to modify his tuple in a way that maintains consistency with the previously answered exact queries. If the distributions of future noisy query answers before and after such modifications are guaranteed to be similar, then Bob can limit further damage[2] – this is a form of plausible deniability.

One simple example of this idea is the relationship between bounded and unbounded differential privacy. Unbounded differential privacy (Definition 1.1) guarantees that the distribution of noisy query answers will change only slightly if Bob removed his tuple from the data. However, if the query "how many tuples are in the data" had been previously answered exactly, then Bob can maintain consistency with this query answer by modifying his tuple arbitrarily (but not deleting it) since the number of tuples would then stay the same. This is the approach taken by *bounded* differential privacy (Definition 1.2), which guarantees that the distribution of future noisy query answers changes only slightly if Bob arbitrarily alters (but not deletes) his tuple.

In the general case, there may be no modification that Bob can make to his tuple that would leave the data consistent with previous query answers. For example, if, for every attribute X, the queries "SELECT X, COUNT(*) GROUP BY X" were exactly answered, then no modification of Bob's tuple alone can leave the data consistent with those query answers (several tuples must be modified simultaneously). However, Bob may collaborate with Alice to jointly change their tuples to maintain consistency. For example, if the number of males and females in the data were released, Bob can change the gender attribute of his tuple to female and Alice can change the gender attribute of her tuple to male to maintain consistency.

Our extension to differential privacy, which accounts for previously released exact query answers, follows these ideas by providing the guarantee that the distribution of future noisy query answers changes only slightly if a small set of tuples were modified in any way that is still consistent with previously answered queries.

4.3 Differential Privacy Subject to Background Knowledge

In this section we modify differential privacy to account for previously released exact query answers. In Section 4.4 we discuss algorithms for privacy-preserving query answering. First, we need the following definitions:

DEFINITION 4.1. (Contingency Table, Cell, Cell count). *Given a table T with attributes $R_1, \ldots, R_d$, a* contingency table *is the (exact) answer to the query:*

```
SELECT R_1, ..., R_d,COUNT(*) FROM T
GROUP BY R_1, ..., R_d
```

A cell *is a specific assignment of values to the attributes (e.g. $R_1 = r_1, \ldots, R_d = r_d$), and a* cell count *is the number of tuples in that cell.*

A contingency table is essentially a table of counts (or a fine-grained histogram).

DEFINITION 4.2. (Move). *Given a contingency table T, a* move *m is a process that adds or deletes a tuple from T, resulting in a contingency table $m(T)$. Thus the value of exactly one cell count changes – it either increases by 1, or decreases by 1 (as long as the result is nonnegative).*

Using this notion of a move, we can define several notions of *neighboring tables*. These different notions will be used with the following definition to allow us to instantiate different versions of differential privacy.

DEFINITION 4.3 (GENERIC DIFFERENTIAL PRIVACY). *A randomized algorithm $\mathcal{A}$ satisfies ϵ-differential privacy if for any set S, $P(\mathcal{A}(T_i)) \leq e^{\epsilon} P(\mathcal{A}(T_j))$ whenever contingency tables T_i and T_j are* ***neighbors****.*

This definition can be specialized into bounded and unbounded differential privacy (Definitions 1.2 and 1.1) through the following choice of neighbors.

- **Unbounded Neighbors** $\mathcal{N}_1$: Contingency tables T_i and T_j are *unbounded neighbors* if one can transform T_i to T_j using only one move. Define the relation $\mathcal{N}_1(T_i, T_j)$ to be true if and only if T_i and T_j are unbounded neighbors.
- **Bounded Neighbors** $\mathcal{N}_2$: Contingency tables T_i and T_j are *bounded neighbors* if the sum of cells of both tables are equal (i.e. T_i and T_j have the same number of tuples) and T_i can be transformed into T_j using exactly 2 moves. Define the relation $\mathcal{N}_2(T_i, T_j)$ to be true if and only if T_i and T_j are neighbors.

Clearly, the use of unbounded neighbors $\mathcal{N}_1$ in Definition 4.3 results in unbounded differential privacy. The use of bounded neighbors $\mathcal{N}_2$ in Definition 4.3 results in bounded differential privacy (compare to Definition 1.2) because changing the value of one tuple from t_1 to t_2 is equivalent to first removing that tuple t_1 (the first move) and then adding t_2 (the second move).

In terms of moves, it is natural to ask why the definition of bounded neighbors $\mathcal{N}_2$ required exactly 2 moves. The reason is that this is the smallest amount of moves that is needed to transform T_i into another table that is consistent with previously released statistics (i.e. the number of tuples in the table, which is assumed to be known in the case of bounded differential privacy).

Another way to think about bounded neighbors $\mathcal{N}_2$ in the context of differential privacy is that T_i and T_j are neighbors if and only if they are as similar as possible, subject to constraints imposed by previously released exact query

[2]Under suitable further assumptions, such as initial independence of records before the release of query answers (See Sections 2 and 3).

answers. Bounded differential privacy then guarantees that an attacker who knows that the true table is either T_i or T_j will have great difficulty distinguishing between these two tables. Note that knowing the true table is either T_i or T_j is the same as knowing $n-1$ tuples in the true database and being unsure about whether the remaining tuple is t_1 or t_2.

4.3.1 Neighbors induced by prior statistics.

Now let us generalize this reasoning to a case where additional exact query answers have been previously released.

EXAMPLE 4.2. *Let T be a contingency table with attributes $R_1, \ldots, R_k$. Due to overriding utility concerns, the following query has already been answered exactly:*

`SELECT GENDER, COUNT(*) FROM T GROUP BY GENDER`

Suppose n is the number of individuals in the table T (i.e. the total cell count), n_F is the number of females and n_M is the number of males. The quantities n_M, n_F, and $n = n_F + n_M$ are now known.

Suppose Drew is in this table. Some limited information about Drew has been disclosed by this query, and now Drew would like a variant of differential privacy to limit any possible damage due to future (noisy) query answers. We seek some form of plausible deniability: the attacker should have difficulty distinguishing between tables T_i and T_j that both have n_F females and n_M males and are otherwise similar except that Drew's tuple takes different values in T_i and T_j.

Thus we need a good choice of neighbors of T (the original table) which should be difficult to distinguish from each other. Unbounded neighbors $\mathcal{N}_1$, used in the definition of unbounded differential privacy (see Definitions 1.1 and 4.3), do not work here; removing or adding a tuple to T changes the number of males and females – thus it is not clear what kind of plausible deniability this would provide. Bounded neighbors $\mathcal{N}_2$, used in the definition of bounded differential privacy (see Definitions 1.1 and 4.3), also do not work; we cannot arbitrarily modify a single tuple and hope to be consistent with the previously released query answers. However, we can make limited changes to 1 or 2 tuples to maintain consistency. For example, we can choose one tuple and modify the attributes other than gender arbitrarily. We can also choose 2 tuples, 1 male and 1 female, swap their gender attributes, and modify the rest of their attributes arbitrarily. Both transformations keep the number of males and females constant. Thus we can provide some level of plausible deniability by making it difficult for an attacker to detect whether one of these transformations has been performed.

This example illustrates 3 important points. First, in the absence of additional correlations (as discussed in Sections 2 and 3), privacy is guaranteed by ensuring that an attacker has difficulty distinguishing between the original table T and a set of neighboring tables that are similar to T. Each neighboring table T' differs from T by a small set of changes (for example, by modifying the value of Drew's tuple and possibly a few others), thus the attacker would have difficulty in detecting these changes (i.e. the attacker would be uncertain about the value of Drew's tuple). Second, each neighboring table T' of T must be consistent with the previously released exact query answers; the reason is that if T' were not consistent, then it would be easy to distinguish between T and T' (since an attacker would know that T' could not possibly be the original table). The third point is that differential privacy can be made aware of prior data release simply by choosing the appropriate notion of neighbors in Definition 4.3 (such modifications have also been suggested by [20]).

	Cell A	Cell B	Cell C	Cell D
Table T_a	4	5	7	3

	Cell A	Cell B	Cell C	Cell D
Table T_b	5	4	6	4

	Cell A	Cell B	Cell C	Cell D
Table T_c	5	4	7	3

Figure 6: Three Contingency Tables

Based on this discussion, we can formally define the concept of neighbors induced by exact query answers (Definition 4.4). Plugging this into generic differential privacy (Definition 4.3), we arrive at the version of differential privacy that takes into account previously released exact query answers.

Before presenting Definition 4.4, first note that if T_a and T_b are contingency tables, then the smallest number of moves needed to transform T_a into T_b is the sum of the absolute difference in cell counts. For example, in Figure 6, this smallest number of moves is 4: first $+1$ for cell A, then $+1$ for cell D, then -1 for cell C, and finally -1 for cell B. Note that any permutation of these moves also transforms T_a into T_b, and these permutations account for all of the shortest sequence of moves (i.e. of length 4) for turning T_a into T_b.

DEFINITION 4.4. (Induced Neighbors $\mathcal{N}_Q$). *Given a set $Q = \{Q_1, \ldots, Q_k\}$ of constraints, let $\mathcal{T}_Q$ be the set of contingency tables satisfying those constraints. Let T_a and T_b be two contingency tables. Let n_{ab} be the smallest number of moves necessary to transform T_a into T_b and let $m_1, \ldots, m_{n_{ab}}$ be one such sequence of moves.*[3] *We say that T_a and T_b are* neighbors induced by Q, *denoted as $\mathcal{N}_Q(T_a, T_b) =$true, if the following holds.*

- *$T_a \in \mathcal{T}_Q$ and $T_b \in \mathcal{T}_Q$.*
- *No subset of $\{m_1, \ldots, m_{n_{ab}}\}$ can transform T_a into some $T_c \in \mathcal{T}_Q$.*

We illustrate this notion of induced neighbors $\mathcal{N}_Q$ (which can now be plugged into generic differential privacy in Definition 4.3) with the following example. Consider contingency table T_a and T_b from Figure 6. Suppose the constraints Q are imposed by our knowledge that the sum of cells A and B is 9 and the sum of cells C and D is 10. We can transform T_a into T_b using the following move sequence: first $+1$ for cell A, then $+1$ for cell D, then -1 for cell C, and finally -1 for cell B. However, using only a subset of these moves, namely $+1$ for cell A and -1 for cell B, we can transform T_a into T_c. Since T_a, T_b and T_c satisfy all of our constraints, T_a and T_b are not neighbors.

Continuing this example, it is easy to see that T_a and T_c are neighbors and T_b and T_c are neighbors (even though T_a and T_b are not). Thus T_a and T_c are as similar as possible while still satisfying constraints imposed by previously released exact query answers (and similarly for T_b and T_c). Thus using this definition of neighbors in generic differential privacy (Definition 4.3), we would be guaranteeing that an

[3]Note that all other such sequences of moves are simply permutations of this sequence.

attacker has difficulty distinguishing between T_a and T_c and also between T_b and T_c.

4.4 Neighbor-based Algorithms for Differential Privacy

In this section we adapt two algorithms for (generic) differential privacy that use the concept of induced neighbors $\mathcal{N}_Q$ (Definition 4.4) to guarantee privacy. We also show that in general this problem (i.e. accounting for previously released exact query answers) is computationally hard.

The first algorithm is an application of the exponential mechanism [26]. To use it, we first need to define a distance function $d(T_a, T_b)$ between two contingency tables. We define the distance function $d(T_a, T_b)$ as follows. Let $\mathcal{T}_Q$ be the set of all contingency tables consistent with previously released query answers. Create a graph where the tables in $\mathcal{T}_Q$ are the nodes, and where there is an edge between tables T_a and T_b if and only if T_a and T_b are induced neighbors. Define $d(T_a, T_b)$ to be the length of the shortest path from T_a to T_b in this graph. The exponential mechanism now works as follows. If T is the true table, we output a table T' with probability proportional to $e^{-\epsilon d(T,T')}$. Following the proof in [26], we can show that this will guarantee 2ϵ-generic differential privacy (Definition 4.3, using $\mathcal{N}_Q$ as the definition of neighbors).

The exponential mechanism [26] generally does not lead to efficient algorithms, even without the complication of induced neighbors $\mathcal{N}_Q$. However, we can modify the Laplace mechanism [8] to suit our purposes. To do this, we first need the concept of *sensitivity*.

DEFINITION 4.5 (SENSITIVITY). *Let $Q = \{Q_1, \ldots, Q_k\}$ be a set of constraints, let $\mathcal{T}_Q$ be the set of contingency tables satisfying those constraints, and let $\mathcal{N}_Q$ be the associated induced neighbors relation. The sensitivity $S_Q(q)$ of a query q over T_Q is defined as:*

$$S_Q(q) = \max_{\mathcal{N}_Q(T_1,T_2)=true} |q(T_1) - q(T_2)| \quad (1)$$

The Laplace mechanism adds Laplace$(S_Q(q)/\epsilon)$ noise (the probability density function is $f(x) = \frac{\epsilon}{2S_Q(q)} e^{-\epsilon x / S_Q(q)}$) to the query answer. The proof of correctness is similar to [8]. Thus we need to be able to compute (an upper bound on) the sensitivity of queries. In some cases (which we describe below) this leads to efficient algorithms, but in general the constraints Q imposed by previously released exact query answers will make the data release problem computationally intractable, as the following results show. We omit detailed proofs due to space constraints.

PROPOSITION 4.1. *Let $Q = \{Q_1, \ldots, Q_k\}$ be a set of constraints. Let S be a table with 3 attributes and let T be the associated contingency table. The following two decision problems are NP-hard in the number of possible attribute values.*

- *Checking whether T has at least one neighbor.*
- *Checking whether two tables T_a, T_b that are consistent with the constraints in Q are* not *neighbors.*

PROPOSITION 4.2. *Given a set $Q = \{Q_1, \ldots, Q_k\}$ of constraints, let $\mathcal{T}_Q$ be the set of contingency tables satisfying those constraints. For any count query q, deciding whether $S_Q(q) > 0$ is NP-hard.*

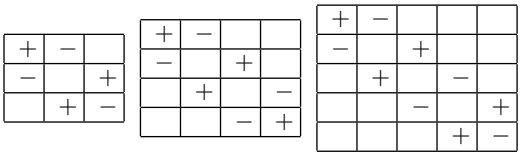

Figure 7:

The above propositions show that the general problem of finding an upper bound on the sensitivity of a query ($S_Q(q) < d$) is at least co-NP-hard, and we suspect that the problem is Π_2^p-complete.

In certain cases, efficient algorithms do exist. For example, consider a 2-dimensional table with attributes R_1 and R_2, and suppose all the row and columns sums are fixed. Let the query q_{all} be defined as:
`SELECT R_1, R_2, COUNT(*) FROM T GROUP BY R_1, R_2.`
The following theorem computes the sensitivity of q_{all}, which can then be answered using the Laplace mechanism.

LEMMA 4.1. *Let Q be the row and column sums for a 2-dimensional $r \times c$ table. The sensitivity $S_Q(q_{all})$ is given by* $\min(2r, 2c)$.

PROOF. (sketch) The key insight is that for a set of moves corresponding to the shortest path from table T_1 to T_2, there can only be exactly 1 tuple added and 1 tuple deleted from cells in the same row or same column. Suppose the contrary: i.e., without loss of generality, there is a designated row where more than 1 tuple is added. Since the row sums are constant, if m tuples are added in a row, then m tuples must have been deleted from cells in the same row. Now suppose we mark every addition to a cell with '+' and a deletion with '-'. Also, if '+' and '-' appear in the same row or column, then draw an edge from the '+' cell to the '-' cell. It is easy to see that if the row and column sums are fixed, then either all or a subset of the marked cells are part of a Hamiltonian cycle alternating in '+' and '-'. If a subset of the marked cells forms the Hamiltonian cycle then we can remove them to get a shorter path from T_1 to T_2 (the cells in the cycle have row and column sums equal to 0, by construction, so their removal preserves the row and column sums). Thus after finitely many such removals we may assume that all of the cells containing a '+' or '-' are part of a Hamiltonian cycle alternating in '+' and '-'. A sequential walk on this loop is a valid set of alternating tuple addition and deletion moves to change T_1 to T_2. This loop enters the designated row for the first time at cell c_1 and exits the designated row for the last time at cell c_2. Cells c_1 and c_2 must have distinct signs. If we drop the part of the loop from c_1 to c_2 and replace it with an edge from c_1 to c_2, we get a new valid cycle from T_1 to T_2. The new cycle is shorter and thus gives a smaller set of moves to change T_1 to T_2; contradicting the fact that the original set of moves was the smallest set of moves. Therefore, each row and column has either no marked cells or exactly one cell marked '+' and one cell marked '-'. Thus the maximum number of moves between any pair of neighbors is at most $\min(2r, 2c)$.

To show that the sensitivity can achieve $\min(2r, 2c)$, we consider pairs of $\ell \times \ell$ tables that differ in the counts as shown (for $\ell = 3, 4, 5$) in Figure 7. It is easy to show pairs of tables for which the smallest set of moves is as described in Figure 7. □

5. RELATED WORK

Differential privacy was developed in a series of papers [1, 10, 8] following the impossibility results of Dinur and Nissim [7] that showed that answering many queries with bounded noise can allow an attacker to reproduce the original database almost exactly. Since then, many additional variations of differential privacy have been proposed [9, 4, 29, 2, 23, 20, 28].

With the success of differentially private algorithms for tabular data [16, 8, 1, 25, 26, 13, 18, 5], there has been interest in developing differentially private mechanisms for other domains, such as social networks [12]. For social networks, examples include [17, 27, 30]. Rastogi et al. [30] proposed a relaxation of differential privacy for social networks by proving an equivalence to adversarial privacy (which makes assumptions about the data), and then adding further constraints on the data-generating mechanism. By the no-free-lunch theorem, the other applications also require assumptions about the data to avoid non-privacy.

Dwork and Naor [8, 11] have several results stating that any privacy definition which provides minimal utility guarantees cannot even safeguard the privacy of an individual whose records are *not collected* by the data curator (because there always exists some auxiliary information that can be combined with query answers to create a threat to privacy). Thus Dwork proposed the principle that privacy definitions should only guarantee privacy for individuals in the data. Kasiviswanathan and Smith [19] formalized the notion of resistance to various attackers including those who know about all but one record in a table and showed an equivalence to differential privacy.

6. CONCLUSIONS

In this paper we addressed several popular misconceptions about differential privacy. We showed that, without further assumptions about the data, its privacy guarantees can degrade when applied to social networks or when deterministic statistics have been previously released. We proposed a principle for evaluating the suitability of a privacy definition to an application: can it hide evidence of an individual's participation in the data generating process from an attacker? When tuples are believed to be independent, deleting one tuple is equivalent to hiding evidence of participation, in which case differential privacy is a suitable definition to use. Providing privacy for correlated data is an interesting direction for future work.

7. REFERENCES

[1] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the SuLQ framework. In *PODS*, 2005.

[2] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *STOC*, 2008.

[3] P. J. Cantwell, H. Hogan, and K. M. Styles. The use of statistical methods in the u.s. census: ütah v. evans*: The American Statistician*, 58(3):203–212, 2004.

[4] K. Chaudhuri and N. Mishra. When random sampling preserves privacy. In *CRYPTO*, 2006.

[5] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. http://arxiv.org/abs/0912.0071.

[6] T. Dalenius. Towards a methodology for statistical disclosure control. *Statistik Tidskrift 15*, 1977.

[7] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS*, 2003.

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

[9] C. Dwork, K. Kenthapadi, F. McSherry, and I. Mironov. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, 2006.

[10] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[11] C. Dwork and M. Naor. On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. *JPC*, 2(1), 2010.

[12] C. Dwork and A. Smith. Differential privacy for statistics: What we know and what we want to learn. *JPC*, 1(2), 2009.

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

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

[15] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis. A framework for efficient data anonymization under privacy and accuracy constraints. *ACM TODS*, 34(2), 2009.

[16] M. Hardt and K. Talwar. On the geometry of differential privacy. In *STOC*, 2010.

[17] M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate estimation of the degree distribution of private networks. In *ICDM*, 2009.

[18] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright. A practical differentially private random decision tree classifier. In *ICDM Workshops*, 2009.

[19] S. P. Kasiviswanathan and A. Smith. A note on differential privacy: Defining resistance to arbitrary side information. http://arxiv.org/abs/0803.3946, 2008.

[20] D. Kifer and B.-R. Lin. Towards an axiomatization of statistical privacy and utility. In *PODS*, 2010.

[21] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *FOCS*, 2000.

[22] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *TKDD*, 1(1), 2007.

[23] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vihuber. Privacy: From theory to practice on the map. In *ICDE*, 2008.

[24] M. Marsili, F. Vega-Redondo, and F. Slanina. The rise and fall of a networked society: A formal model. *PNAS*, 101(6), 2004.

[25] F. McSherry and I. Mironov. Differentially private recommender systems: building privacy into the net. In *KDD*, 2009.

[26] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, 2007.

[27] D. J. Mir and R. N. Wright. A differentially private graph estimator. In *ICDM Workshops*, 2009.

[28] I. Mironov, O. Pandey, O. Reingold, and S. Vadhan. Computational differential privacy. In *CRYPTO*, 2009.

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

[30] V. Rastogi, M. Hay, G. Miklau, and D. Suciu. Relationship privacy: Output perturbation for queries with joins. In *PODS*, 2009.

[31] P. Samarati. Protecting respondents' identities in microdata release. *TKDE*, pages 1010 – 1027, 2001.

[32] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 1965.