Background Knowledge in Differential Privacy: Exact Cuboids in a Noisy Data Cube

ABSTRACT

This is the abstract...

1. INTRODUCTION

Since the notion of *differential privacy* was introduced in [4, 2], it has become more and more popular recently. As indicated by its name, it protects privacy of any differential user, or in other words, including into or excluding any particular user from the database does not affect the output of a differentially private algorithm to a sensible degree. However, as shown in [6], certain types of background knowledge may break differential privacy. An undesirable consequence is that, with certain background knowledge, an adversary could reduce the noise variable by a factor proportional to the size of the dataset.

Organization. In this paper, we first review the notation of *generic differential privacy* [6] in Section 2. This notation of privacy aims to incorporate adversary's background knowledge into the model of differential privacy. In the contexts of data cubes, we are interested in background knowledge in the form of some cuboids that need to be released exactly (because of laws of policies). Adversaries can utilize these exact cuboids to attack individuals' privacy. Intuitively, the more background knowledge adversaries have, the more noise needs to be injected to preserve individuals' privacy. By studying generic differential privacy, we want to answer questions about how much noise is necessary for certain amount/type of adversary's background knowledge. We introduce new results on how to calibrate noise to be injected for preserving generic differential privacy and its properties for tabular data Section 4 and for graph data Section 5. We then show how to enforce consistency in noisy cuboids w.r.t. background knowledge in Section 6. Intuitively, for adversaries with more background knowledge, we have to insert more noise initially, but those exact cuboids can in turn help reduce noise in the consistency-enforcing process. Applications and experiments of the proposed techniques are introduced in Section 7. We finally conclude this paper with related works in Section 8 and conclusions in Section 9.

2. BACKGROUND AND NOTATIONS

In this section, we first introduce *data cubes*, a modeling and analytical tool for multidimensional data. We then introduce some basic concepts of *differential privacy*, which was originally introduced in [2, 4]. In order to handle exact background knowledge (or exact cuboids) owned by adversaries, we review the definition of *generic differential privacy* [6] in the third part.

2.1 Data Cubes

A *fact table* T is a multidimensional table with d *nominal dimensions* $\mathcal{A} = \{A_1, A_2, \ldots, A_d\}$. We use A_i to denote both the i^{th} nominal dimension and the set of all possible values for this dimension. Let $|A_i|$ denote the number of distinct values (i.e., *cardinality*) of dimension A_i. A fact table T is a set of rows, and for each row $r \in T$, $r[i]$ is r's value for A_i.

The *data cube* [5] of a fact table T can be considered as the projections of T onto subset of dimensions. A data cube consists of *cells* and *cuboids*. More formally, a cell a takes the form $(a[1], a[2], \ldots, a[d])$, where $a[i] \in (A_i \cup \{*\})$ denotes the i^{th} dimension's value for this cell, and it is associated with certain aggregate measure of interest. In this paper, we focus on the *count measure* c. The count measure $\mathsf{c}(a)$ is the number of rows r in T that are aggregated in cell a (with the same values on non-$*$ dimensions of a). Formally, let $\mathsf{c}(a) = |\{r \in T \mid \forall 1 \le i \le d,\ r[i] = a[i] \vee a[i] = *\}|$.

A cell a is an *m-dim cell* if values of exactly m dimensions $a[i_1], \ldots, a[i_m]$ are *not* $*$. An *m-dim cuboid* C is specified by m dimensions $[C] = \{A_{i_1}, A_{i_2}, \ldots, A_{i_m}\}$, and it consists of all m-dim cells a such that $\forall 1 \le k \le m$, $a[i_k] \in A_{i_k}$, and $a[j] = *$ for all $j \notin \{i_1, i_2, \ldots, i_m\}$. We use C and $[C]$ interchangeably to denote either a

Sex	Age	Salary
F	21-30	10-50k
F	21-30	10-50k
F	31-40	50-200k
F	41-50	500k+
M	21-30	10-50k
M	21-30	50-200k
M	31-40	50-200k
M	60+	500k+

9(a) Fact Table T

Sex	Age	Salary	c
$*$	$*$	0-10k	0
$*$	$*$	10-50k	3
$*$	$*$	50-200k	3
$*$	$*$	500k+	2

9(b) Cuboid {Salary}

Sex	Age	Salary	c
$*$	21-30	$*$	4
$*$	31-40	$*$	2
$*$	41-50	$*$	1
$*$	60+	$*$	1

9(c) Cuboid {Age}

Sex	Age	Salary	c
$*$	21-30	0-10k	0
$*$	21-30	10-50k	3
$*$	21-30	50-200k	1
$*$	21-30	200-500k	0
$*$	21-30	500k+	0
$*$	31-40	0-10k	0
$*$	31-40	10-50k	0
$*$	31-40	50-200k	2
$*$	31-40	200-500k	0
$*$	31-40	500k+	0
$*$	...	...	...

9(d) Cuboid {Age, Salary}

Figure 1: A fact Table and its count data cube

cuboid or the set of dimensions specifying this cuboid. A cuboid C can be interpreted as the projection of T on the set of dimensions $[C]$. The d-dim cuboid is called the *base cuboid* and the cells in it are *base cells*.

For two cuboids C_1 and C_2, if $[C_1] \subseteq [C_2]$ (denoted as $C_1 \preceq C_2$), *count* measure c of cells in C_1 can be computed from C_2. The cuboid C_1 is said to be an *ancestor* of C_2, and C_2 is a *descendant* of C_1. Let $\mathcal{L}_{\mathsf{all}}$ denote the set of all cuboids. $(\mathcal{L}_{\mathsf{all}}, \preceq)$ forms a *lattice*.

Example 2.1. *Fact table T in Table 1 has three dimensions:* Sex = {M, F}*;* Age = {0-10, 11-20, 21-30, 31-40, 41-50, 51-60, 60+}*; and* Salary = {0-10k, 10-50k, 50-200k, 200-500k, 500k+}*. Cuboid* {Age, Salary} *can be computed from T. For example, to compute the cell* ($*$, 21-30, 10-50k)*, we refer to the table T for rows* (M, 21-30, 10-50k) *and* (F, 21-30, 10-50k)*, and we find three such rows. As* {Salary} $\subseteq$ {Age, Salary}*, cuboid* {Salary} *can be computed from cuboid* {Age, Salary}*. For example, to compute cell* ($*$, $*$, 10-50k)*, we aggregate cells* ($*$, 0-10, 10-50k)*, ..., * ($*$, 60+, 10-50k).

2.2 Differential Privacy

Differential privacy is based on indistinguishability between pairs of neighboring tables. In this notation of privacy, wwo tables T_1 and T_2 are *neighbors* if they differ by one row, i.e., inserting or deleting one row from T_1 yields T_2. Formally, let $\mathcal{T}$ be the set of all possible table instances with dimensions $A_1, A_2, \ldots, A_d$, and $\Gamma(T) = \{T' \mid |(T-T') \cup (T'-T)| = 1\} \subseteq \mathcal{T}$ the set of all neighbors of T. Clearly, $T_1 \in \Gamma(T_2)$ implies $T_2 \in \Gamma(T_1)$.

Definition 2.1. *(Differential privacy [3]) A randomized algorithm $\mathcal{K}$ is ϵ-differentially private if for any two neighboring tables T_1 and T_2 and any subset S of the output of $\mathcal{K}$,*

$$\Pr[\mathcal{K}(T_1) \in S] \leq \exp(\epsilon) \times \Pr[\mathcal{K}(T_2) \in S],$$

where the probability is taken over the randomness of $\mathcal{K}$ and ϵ is a fixed value.

Consider an individual's concern about the privacy of her/his record r. When the publisher specifies a small ϵ, Definition 2.1 ensures that the inclusion or exclusion of r in the fact table makes little difference in the chances of $\mathcal{K}$ returning any particular answer.

Some papers use a slightly different definition of neighbors: T_1 and T_2 are neighbors if they have the same cardinality and T_1 can be obtained from T_2 by replacing one row (ϵ-indistinguishable [4]). Our algorithms and techniques in this thesis also work with this definition, if we double the noise, because it has been shown that any mechanism satisfies ϵ-indistinguishability if and only if it satisfies 2ϵ-differential privacy.

There are two popular ways to achieve ϵ-differential privacy, the *Laplacian mechanism* [3] and the *exponential mechanism* [9], both of which are based on the *sensitivity* of a publishing function F. Throughout this paper, we use $\mathcal{R}$ to denote the set of all real numbers. Let $F : \mathcal{T} \to \mathcal{R}^n$ be a function (or a query sequence) that produces a vector of length n from a table instance.

Definition 2.2. *(Sensitivity [3]) The L^1 global sensitivity of F is:*

$$S(F) = \max_{T_2 \in \mathcal{T}} \left(\max_{T_1 \in \Gamma(T_2)} \|F(T_1) - F(T_2)\|_1 \right),$$

where $\|x - y\|_1 = \sum_{1 \leq i \leq n} |x_i - y_i|$ is the L^1 distance between two n-dimensional vectors $x = \langle x_1, x_2, \ldots, x_n \rangle$ and $y = \langle y_1, y_2, \ldots, y_n \rangle$.

For example, if the function F is to count how many rows satisfying property P, then $S(F) = 1$, because deleting/adding one row in T changes the count $F(T)$ by at most one.

Definition 2.3. *(Laplacian distribution) Let* $\mathrm{Lap}(\lambda)$ *denote a sample Y taken from a zero-mean Laplace distribution with the probability density function $h(x) = \frac{1}{2\lambda}\exp(-|x|/\lambda)$. We have $\Pr[|Y| \geq z] = \exp(-z/\lambda)$, $\mathrm{E}[Y] = 0$, and variance $\mathrm{Var}[Y] = 2\lambda^2$. We write $\langle \mathrm{Lap}(\lambda) \rangle^n$ to denote a vector of n independent random samples $\langle Y_1, Y_2, \ldots, Y_n \rangle$, where $Y_i \sim \mathrm{Lap}(\lambda)$.*

Laplacian mechanism is a class of differentially private algorithms which output $F(T)$ with random noise

1).

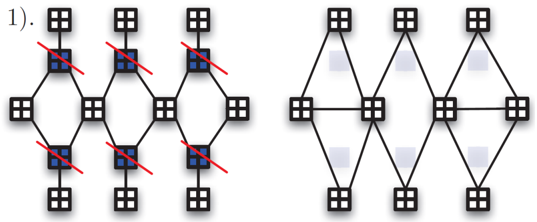

Figure 2: Neighboring Tables and Constraints

injected. The following theorem quantifies how much noise to be injected into $F(T)$ is sufficient.

THEOREM 2.1. *(Laplacian mechanism [4]) Let $F : \mathcal{T} \to \mathcal{R}^n$ be a query sequence of length n against a table $T \in \mathcal{T}$. The randomized algorithm that takes as input table T and output $\tilde{F}(T) = F(T) + \langle \text{Lap}(S(F)/\epsilon) \rangle^n$ is ϵ-differentially private.*

3. ASHWIN PRIVACY SECTION

Kifer and Machanavajjhala [7] showed that in the presence of exact count constraints about the database, a differentially private mechanism does not limit the ability of an adversary to recover cell counts from the base table with high accuracy.

EXAMPLE 3.1. *Consider a fact table D with one attribute R with a domain $\{r_1, r_2, \ldots, r_k\}$. Let the cell count corresponding to value r be denoted by $c(r)$. Suppose, based on publicly released datasets the following $k-1$ constraints are already known: $\forall i, c(r_i)+c(r_{i+1}) = a_i$. Since there are k unknown counts and $k-1$ equations, adversary can't reconstruct the table. However, if we knew one of the $c(r_i)$, then the entire table can be reconstructed – in this way the tuples are correlated.*

According to differential privacy, each count $c(r_i)$ can be answered by adding independent noise drawn from a Laplace distribution with parameter $(1/\epsilon)$ (variance $= 2/\epsilon^2$). While these noisy counts $\tilde{c}(r_i)$ do not disclose information about any individual, they can be combined with the constraints to get very precise estimated of $c(r_i)$. For instance, we can construct k independent estimators for $c(r_1)$ based on $\tilde{c}(r_1)$, $a_1 - \tilde{c}(r_2)$, $a_1 - a_2 + \tilde{c}(r_3)$ and so on, each having a variance of $2/\epsilon^2$. By averaging these estimators, we can predict the value of $c(r_i)$ with a variance of $2/(k\epsilon^2)$. For large k (e.g., when there are 2^d values in R), the variance is so small that D can reconstructed exactly with high probability.

The reason behing the attack can be explained as follows.

Figure 2 explains the intuition behind why differential privacy does not compose with deterministic constraints, and motivates the privacy definition we use in this paper. The figure on the left shows a set of all possible datasets in the absence of constraints; edges are drawn whenever two datasets are neighboring, i.e., differ in one record. Differential privacy ensures that an adversary can't distinguish between any pair of neighboring datasets. However, when marginals are known, some of the datasets are not possible, as they do not satisfy the constraints. For instance, if it is publicly known that there are 10 men in the data, changing the gender of one of the male records to female results in a dataset with 9 men (which is inconsistent with the known constraint and hence impossible). In the figure, for every pair of neighboring datasets, one of them is not possible. It is possible that an adversary may now be able to distinguish between the new neighboring datasets (on the right). Therefore, one must ensure that an adversary can't distinguish datasets that are induced neighbors after accounting for the constraints.

In this section, we present privacy definition that extend differential privacy by ensuring that the adversary can't distinguish between pairs of *induced neighbors* – tables that satisfy the publicly known constraints, but are minimally different. We present three definitions for induced neighbors – *Markov bases*, *positive induced neighbors*, and *worst case induced neighbors* – and thus provide three extensions to differential privacy which vary in the amount of privacy protection guaranteed. We explain these notions in detail next.

Need to copy over the some of the definitions from oldprivacy.tex...

3.1 Markov Bases

Our first notion of neighbors is motivated by the concept of Markov Bases from statistics [1].

DEFINITION 3.1. *Two tables T_1, T_2 are considered Markov Bases neighbors with respect to a pre-released set of cuboids $\mathcal{C}$ if:*

- *Both T_1 and T_2 satisfy $\mathcal{C}$,*
- *If C_1 and C_2 represent the base cuboids of T_1 and T_2 respectively, then $C_1 - C_2$ has non zero counts in at most 4 cells, and the non zero counts are either 1 or -1.*

In particular, we can easily show that C_1 and C_2 from the above definition differ in exactly one of two ways:

- Consider some tuple $(a_1, a_2, \ldots, a_i, \ldots, a_d)$. Suppose attribute i does not appear in a pre-released cuboids. Then changing a_i to a'_i decreases the count of $c_1 = (a_1, a_2, \ldots, a_i, \ldots, a_d)$ by 1 and increases the count of $c_2 = (a_1, a_2, \ldots, a'_i, \ldots, a_d)$ by 1.
- Suppose attributes i and j appear in two different cuboids. Then changing a_i to a'_i requires changing some other tuple that contains a'_i (say $(a'_1, a'_2, \ldots, a'_d)$) to a_i. Hence, at most four cell counts are non zero in $C_1 - C_2$, and the non-zero counts are either 1 or -1.

	a_1	a_2	a_3	$C(B)$
b_1	1	0	0	1
b_2	0	1	0	1
b_3	0	0	1	1
$C(A)$	1	1	1	

	a_1	a_2	a_3	$C(B)$
b_1	0	1	0	1
b_2	0	0	1	1
b_3	1	0	0	1
$C(A)$	1	1	1	

Figure 3: Worst case induced neighboring tables T and T' from Example 3.3. (a_1, b_2) is colored red, since the adversary knows that such a tuple does not exist.

EXAMPLE 3.2. *Consider the fact table T in Figure 1(a), and suppose the cuboids $C\{\mathsf{Age}\}(T)$ and $C\{\mathsf{Salary}\}(T)$ has been pre-released. Consider T_1 where value for* Sex *attribute in the first tuple is* M. *T_1 is a Markov basis neighbor. Also, consider T_2 where the* Age *values of tuples 3 and 4 are swapped. T_2 is also a Markov basis neighbor.*

As is evident from the examples, the differential privacy definition based on Markov basis induced neighbors ensures that an attacker can't distinguish between two datasets where one individual's tuple changes in *one attribute* based on the output of the mechanism. We can show that this definition implicitly makes an assumption that adversaries do not know any correlations between attributes. Due to space constraints we omit the formal analysis of this claim and defer it to the full version of the paper.

3.2 Worst Case Induced Neighbors

We next describe a worst case definition for induced neighbors. This definition was first used by Kifer and Machanavajjhala [7].

Copy from oldprivacy.tex ...

EXAMPLE 3.3. *Consider a 2-D table with two attributes $A = \{a_1, a_2, a_3\}$ and $B = \{b_1, b_2, b_3\}$, with the A and B cuboids (row and column sums) published (see Figure 3). Let T be a table with tuples $(a_1, b_1), (a_2, b_2)$ and (a_3, b_3). This corresponds to the cuboid (on the left) in Figure 3. Suppose the adversary knows that (a_1, b_2) can not exist in the table (e.g., a structural zero in statistical tables). Then if the first tuple changes its value from (a_1, b_1) to (a_2, b_1), then the only table that satisfies the $C(A)$ and $C(B)$ cuboids is $T' = (a_2, b_1), (a_3, b_1), (a_1, b_3)$. Hence, T and T' are worst case induced neighbors.*

3.3 Positive Induced Neighbors

Finally, we describe an third induced neighbor definition which provides stronger protection than Markov basis induced neighbors, but weaker protection than worst case induced neighbors.

need a formal definition

EXAMPLE 3.4. *Consider a 3-D table with three attributes $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$, and $C = \{c_1, c_2, c_3\}$ with the A, B and C cuboids published. Let T be a table with tuples $(a_1, b_1, c_1), (a_2, b_2, c_2)$ and (a_3, b_3, c_3). Suppose the first tuple changes its value from (a_1, b_1, c_1) to (a_1, b_2, c_3). In order to satisfy the pre-released A, B and C cuboids, T' must include tuples $(a_2, b_3, c_1), (a_3, b_1, c_2)$ other than (a_1, b_2, c_3). If there are additional tuples in T, then these tuples can remain unchanged in T'. Hence, T and T' are positive induced neighbors.*

3.4 Discussion

The following lemma shows that for a table T, the set of Markov basis induced neighbors is a subset of the positive induced neighbors, which in turn is a subset of the worst case induced neighbors. Therefore, these three definitions lead to variants of differential privacy that provide increasing levels of privacy protection (i.e., perturbation) and thus decreasing amounts of utility for the same query.

prove the lemma ...

talk about sensitivity in the next section.

4. CALIBRATING NOISE IN TABLES

We need to compute generic sensitivity $S_{\mathcal{Q}}$ in order to apply the generic Laplacian mechanism, and [6] shows that it is hard to estimate the generic sensitivity for general data type and background knowledge. In Section 4.1, we will show that it is possible to estimate $S_{\mathcal{Q}}$ in data cubes when the background knowledge is one exact cuboid or a set of two exact cuboids (Definition **??**). In Section 4.2, we will introduce two useful properties: *projection* and *composition*.

4.1 Estimating Generic Sensitivity

Consider a d-dim fact table T with attributes $\mathcal{A} = \{A_1, A_2, \ldots, A_d\}$. Recall $|A_i|$ is the cardinality of dimension A_i. Let C_{base} be the d-dim base cuboid. For a cuboid C, recall $[C]$ is the set of its dimensions. For a subset of dimensions $\mathcal{A}' \subseteq \mathcal{A}$, let $\text{size}(\mathcal{A}') = \prod_{A_i \in \mathcal{A}'} |A_i|$. Define $\text{size}(\emptyset) = 1$. $\text{size}([C])$ is essentially the number of all possible cells in cuboids C.

Kifer and Machanavajjhala in [6] study a special case for 2-dim fact tables.

LEMMA 4.1 ([6]). *Consider a 2-dim fact table T with two attributes A_1 and A_2. (i) If the background knowledge $\mathcal{Q} = \{\{A_1\}\}$ (row sums or column sums), the generic sensitivity of publishing C_{base} is $S_{\mathcal{Q}}(C_{\mathsf{base}}) = 2$. (ii) If $\mathcal{Q} = \{\{A_1\}, \{A_2\}\}$ (both row sums and column sums), the generic sensitivity of publishing C_{base} is $S_{\mathcal{Q}}(C_{\mathsf{base}}) = 2\min\{|A_1|, |A_2|\}$.*

We now generalize Lemma 4.1 for d-dim fact table T, with background knowledge $\mathcal{Q}$ as any one or two cuboids. Again we want to publish the base cuboid C_{base}, subject to $\mathcal{Q}$.

LEMMA 4.2. *Consider a d-dim fact table T. (i) If the background knowledge $\mathcal{Q} = \{C\}$ for any cuboid C,* *the generic sensitivity of publishing* C_{base} *is* $S_{\mathcal{Q}}(C_{\mathsf{base}}) = 2$. *(ii) If* $\mathcal{Q} = \{C_1, C_2\}$ *for any two cuboids, then* $S_{\mathcal{Q}}(C_{\mathsf{base}}) = 2\min\{\text{size}([C_1]-[C_2]), \text{size}([C_2]-[C_1])\}$.

Proof. The proof for (i) is trivial and is very similar to the proof for (i) in Lemma 4.1.

For (ii), let's first prove $S_{\mathcal{Q}}(C_{\mathsf{base}}) \le 2\min\{\text{size}([C_1]-[C_2]), \text{size}([C_2]-[C_1])\}$.

Recall $\mathcal{T}$ is the set of all d-dim fact tables with attributes $\mathcal{A} = \{A_1, A_2, \ldots, A_d\}$, and $\mathcal{T}_{\mathcal{Q}} \subseteq \mathcal{T}$ is the set of all feasible fact tables subject to background knowledge $\mathcal{Q}$. Consider any two feasible fact tables $T_a, T_b \in \mathcal{T}_{\mathcal{Q}}$, it suffices to show that, if a shortest sequence M_{ab} of moves that transforms T_a into T_b is longer than $2\min\{\text{size}([C_1]-[C_2]), \text{size}([C_2]-[C_1])\}$, a subsequence of these moves can transform T_a into another feasible table $T_c \in \mathcal{T}_{\mathcal{Q}}$. Then, for any two (induced) neighboring fact table T_a and T_b, we must have the length of the shortest move sequence $|M_{ab}| \le 2\min\{\text{size}([C_1]-[C_2]), \text{size}([C_2]-[C_1])\}$, and thus we can have $S_{\mathcal{Q}}(C_{\mathsf{base}}) \le |M_{ab}| \le 2\min\{\text{size}([C_1]-[C_2]), \text{size}([C_2]-[C_1])\}$, because the L^1 distance between the base cuboids on the two tables T_a and T_b is no more than $|M_{ab}|$.

For a cell a in C_{base}, and the two cuboids C_1 and C_2, recall $a[C_i]$ is the projection of a on $[C_i]$, i.e., the cell in C_i which has the same attributes values as a in $[C_i]$.

We construct a directed bipartite graph as follows. The vertices are cells in C_1 and C_2. In M_{ab}, for a move which increases the count $\mathsf{c}(a)$ of a cell a in C_{base} by one (*positive move*), we create an edge from $a[C_1]$ to $a[C_2]$; and for a move which decreases the count $\mathsf{c}(a)$ by one (*negative move*), we create an edge from $a[C_2]$ to $a[C_1]$. Because both T_a and T_b are feasible, we have the same number of positive moves incident on $a[C_i]$ as the number of negative moves; or, in this graph, each vertex has the same in-degree as the out-degree.

A directed sub-cycle in this graph corresponds to a subset of the moves which transform T_a into another feasible table T_c (because one positive move and one negative move incident on cells of C_1 and C_2 do not violate the background knowledge). So we want to show that, if $|M_{ab}| > 2\min\{\text{size}([C_1]-[C_2]), \text{size}([C_2]-[C_1])\}$, there exists such a nontrivial proper sub-cycle, which contracts to the fact that T_a and T_b are induced neighbors.

Indeed, if $|M_{ab}| > 2\min\{\text{size}([C_1]), \text{size}([C_2])\}$, there exists such a sub-cycle. There is an Eulerian directed walk W (starting and ending at the same vertex) in this graph, because each vertex has the same in-degree as out-degree. $|M_{ab}| > 2\min\{\text{size}([C_1]), \text{size}([C_2])\}$ implies that some vertex v in C_1 or C_2 has degree at least 4, and thus it is visited at least twice in W (i.e. $W = u \ldots vW'v \ldots u$). So we can extract a nontrivial proper sub-cycle from W at vertex v (it could be $vW'v$).

To get the tight bound $|M_{ab}| \le |2\min\{\text{size}([C_1]-[C_2]), \text{size}([C_2]-[C_1])\}$, we note that the graph has been divided into $\text{size}([C_1] \cap [C_2])$ connected components in nature. Because, according to the construction, any two cells $a_1 \in C_1$ and $a_2 \in C_2$ are connected only if a_1 and a_2 have the same values in the attributes $[C_1] \cap [C_2]$. And in each component, we have $\text{size}([C_1]-[C_2])$ vertices from C_1 and $\text{size}([C_2]-[C_1])$ vertices from C_2 (and each vertex has the same in-degree as the out-degree). So based on an argument similar to the one for $|M_{ab}| \le 2\min\{\text{size}([C_1]), \text{size}([C_2])\}$, we can prove this tight bound as well.

To prove $S_{\mathcal{Q}}(C_{\mathsf{base}})$ is indeed equal to $2\min\{\text{size}([C_1]-[C_2]), \text{size}([C_2]-[C_1])\}$, we construct a cycle with length $2\min\{\text{size}([C_1]-[C_2]), \text{size}([C_2]-[C_1])\}$, which corresponds to a shortest sequence M_{ab} of moves between two feasible tables. It satisfies two properties: (i) the base cuboids on the two tables have L^1 distance equal to $|M_{ab}|$; and (ii) no subset of M_{ab} can yield another feasible table. It can be constructed as follows: Number all possible assignments of values to attributes in $[C_1]-[C_2]$ as $1, 2, \ldots, \text{size}([C_1]-[C_2])$, and all possible assignments of values to attributes in $[C_2]-[C_1]$ as $1, 2, \ldots, \text{size}([C_2]-[C_1])$. Consider such a cycle: $(1, x) \to (x, 1) \to (2, x) \to (x, 2) \to \ldots \to (1, x)$, where (i, x) is a cell in C_1 with value i in attributes $[C_1]-[C_2]$ and value x in $[C_1] \cap [C_2]$, and similarly (x, j) is a cell in C_2 with j in $[C_2]-[C_1]$ and x in $[C_1] \cap [C_2]$. This cycle has length $2\min\{\text{size}([C_1]-[C_2]), \text{size}([C_2]-[C_1])\}$. Easy to very this cycle satisfies properties (i) and (ii). So the proof is completed. $\square$

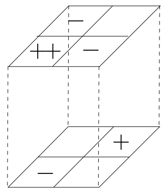

Figure 4: Shortest move sequence for 2x2x2 case

Note that Lemma 4.2 also considers a special case: when two exact cuboids $C_1 \preceq C_2$, $S_{\mathcal{Q}}(C_{\mathsf{base}}) = 2$.

Example 4.1. *Consider the fact table* T *in Table 1 with three dimensions:* Sex, Age, *and* Salary. *Suppose two cuboids* $C_1 = \{\mathsf{Sex}, \mathsf{Age}\}$ *and* $C_2 = \{\mathsf{Age}, \mathsf{Salary}\}$ *are to be released exactly (*$\mathcal{Q} = \{C_1, C_2\}$*). From Lemma 4.2, the sensitivity of releasing* $C_{\mathsf{base}} = \{\mathsf{Sex}, \mathsf{Age}, \mathsf{Salary}\}$ *subject to* $\mathcal{Q}$ *is* $2\min\{\text{size}([C_1]-[C_2]), \text{size}([C_2]-[C_1])\} = 2\min\{|\mathsf{Sex}|, |\mathsf{Salary}|\} = 4$.

When the number of cuboids $|\mathcal{Q}| \ge 3$, the problem of estimating the sensitivity $S_{\mathcal{Q}}(C_{\mathsf{base}})$ is quite open. We only have results for some special cases. For example, when $\mathcal{Q} = \{\{A_1\}, \{A_2\}, \{A_3\}\}$, where the cardinalities of attributes are $|A_1| = 2$, $|A_2| = 2$, and $|A_3| = x$: if $x = 2$ or 3, we have $S_{\mathcal{Q}}(C_{\mathsf{base}}) = 6$ (refer to Figure 4), and if $x = 4$, we have $S_{\mathcal{Q}}(C_{\mathsf{base}}) = 8$.

Lemma 4.2 can be extended to the case when the adversary has partial knowledge of the two cuboids in $\mathcal{Q}$, or in other words, only some cells in the two cuboids are published exactly. We will show that the amount of noise needed to be inserted is linear proportional to the number of exact cells when they are from two cuboids.

Definition 4.1. *(General Background Knowledge) In a data cube with d dimensions $\{A_1, A_2, \ldots, A_d\}$, general background knowledge $\mathcal{Q}$ is a set of cells (from one or more cuboids) that are to be released exactly.*

Lemma 4.3. *Consider a d-dim fact table T. (i) If the background knowledge $\mathcal{Q} \subseteq C$ for any cuboid C, the generic sensitivity of publishing C_{base} is $S_{\mathcal{Q}}(C_{\mathsf{base}}) = 2$. (ii) If $\mathcal{Q} \subseteq C_1 \cap C_2$ for any two cuboids, then*

$$S_{\mathcal{Q}}(C_{\mathsf{base}}) = 2 \max_x \min \left\{ \begin{array}{l} |\{a_1 \mid a_1[C_1 \wedge C_2] = x\}|, \\ |\{a_1 \mid a_1[C_1 \wedge C_2] = x\}| \end{array} \right\}.$$

Proof. To be added... $\square$

4.2 Projection and Composition

There are two useful properties of generic differential privacy, *projection* (for tabular data) and *composition*.

The *projection* property is about how much the background knowledge is projected into a subspace (cuboid) of the fact table. Note that Lemma 4.2 is about the sensitivity of releasing the base cuboid C_{base} subject to the background knowledge $\mathcal{Q}$. A natural question is: subject to $\mathcal{Q}$, how much is the sensitivity of releasing any cuboid C? The following lemma says that, the sensitivity stays almost unchanged for all the cuboids C (except the ones that can be computed from $\mathcal{Q}$), if the background knowledge $\mathcal{Q}$ is fixed.

Lemma 4.4. *(Projection property) In a data cube, if the background knowledge $\mathcal{Q}$ is fixed, for any cuboid C that cannot be computed from $\mathcal{Q}$, we have $S_{\mathcal{Q}}(C) \leq S_{\mathcal{Q}}(C_{\mathsf{base}})$. If $[C] - \bigcup_{C_i \in \mathcal{Q}}[C_i] \neq \emptyset$, $S_{\mathcal{Q}}(C) = S_{\mathcal{Q}}(C_{\mathsf{base}})$; otherwise, $\frac{1}{2}S_{\mathcal{Q}}(C_{\mathsf{base}}) \leq S_{\mathcal{Q}}(C) \leq S_{\mathcal{Q}}(C_{\mathsf{base}})$*

Proof. $S_{\mathcal{Q}}(C) \leq S_{\mathcal{Q}}(C_{\mathsf{base}})$ is obvious. For any two induced neighbors T_a and T_b, let C^a and C^b be the cuboids constructed from T_a and T_b, respectively, on dimensions $[C] = [C^a] = [C^b]$. Similarly for C^a_{base} and C^b_{base}. Then we must have $\|C^a - C^b\|_1 \leq \|C^a_{\mathsf{base}} - C^b_{\mathsf{base}}\|_1$.

If there is an attribute A_0 in $[C] - \bigcup_{C_i \in \mathcal{Q}}[C_i]$, consider any two induced neighbors T_a, T_b, and a shortest sequence M_{ab} of moves that transforms T_a into T_b. A_0 has at least two values, say, x and y. Then for each positive move in M_{ab}, we can enforce it add a tuple with $A_0 = x$, and for each negative move in M_{ab}, we can enforce it delete a tuple with $A_0 = y$. In other words, positive moves and negative moves do not cancel out each other in C. So we must have $\|C^a - C^b\|_1 = |M_{ab}| = \|C^a_{\mathsf{base}} - C^b_{\mathsf{base}}\|_1$, and thus $S_{\mathcal{Q}}(C) = S_{\mathcal{Q}}(C_{\mathsf{base}})$.

If $[C] - \bigcup_{C_i \in \mathcal{Q}}[C_i] = \emptyset$, we can carefully arrange the positive moves and negative moves in M_{ab} within C, so that most of them do not cancel out each other. In most cases, we still can show that $S_{\mathcal{Q}}(C) = S_{\mathcal{Q}}(C_{\mathsf{base}})$, and in the worst case, $S_{\mathcal{Q}}(C) \geq \frac{1}{2}S_{\mathcal{Q}}(C_{\mathsf{base}})$. $\square$

The *transition* and *composition* properties allow the combining of the publications or outcomes of several differentially private algorithms [8]. These two properties still hold for generic differential privacy as long as the background knowledge $\mathcal{Q}$ is fixed. The statement and the proof are similar. We repeat the statement for completeness:

Theorem 4.5. *(Transition and composition properties) Let $\mathcal{K}_i(\cdot)$ or $\mathcal{K}_i(\cdot, \cdot)$ be a randomized algorithm which is ϵ_i-differentially private subject to background knowledge $\mathcal{Q}$.*

1. *For a table T, outputting $\mathcal{K}_1(T)$ and $\mathcal{K}_2(T, \mathcal{K}_1(T))$ is $(\epsilon_1 + \epsilon_2)$-differentially private subject to background knowledge $\mathcal{Q}$.*
2. *For any table T and any algorithm $\mathcal{K}$, outputting $\mathcal{K}_1(T)$ and $\mathcal{K}(\mathcal{K}_1(T))$ is ϵ_1-differentially private subject to background knowledge $\mathcal{Q}$.*
3. *For any two tables T_1 and T_2 such that $T_1 \cap T_2 = \emptyset$, outputting $\mathcal{K}_1(T_1)$ and $\mathcal{K}_1(T_2)$ is ϵ_1-differentially private subject to background knowledge $\mathcal{Q}$.*

5. CALIBRATING NOISE IN GRAPHS

The results presented in the last section can be extended for graph data. We consider node degrees as background knowledge (e.g., numbers of friends in Facebook are possible known to everyone). Let $G(V, E)$ be a graph with node set V and edge set E. For each node $v \in V$, let $d(v)$ be its degree (i.e., number of edges incident on v) if G is an undirected graph; and let $d^+(v)$ and $d^-(v)$ be its in-degree and out-degree, respectively, if G is a directed graph. To convert a graph $G(V, E)$ into a table, consider such a table T with two dimensions V_1 and V_2: for each edge $(v, u) \in E$, create a row in the table T, with dimension V_1 values to v and dimension V_2 values to u. Degrees and in/out-degrees are column sums of this table. We have the following results.

Lemma 5.1. *There are two cases: (i) Consider an undirected graph $G(V, E)$. If the background knowledge is $\mathcal{Q} = \{d(v_1), d(v_2), \ldots, d(v_n)\}$, i.e., degrees of n nodes, the generic sensitivity of publishing the graph is*

$$S_{\mathcal{Q}}(G) = \min\{n + 1, |V|\}.$$

(ii) Consider a directed graph $G(V, E)$. If the background knowledge is $\mathcal{Q} = \{d^+(v_1), d^+(v_2), \ldots, d^+(v_{n_1})\} \cup \{d^-(u_1), d^-(u_2), \ldots, d^-(u_{n_2})\}$, i.e., in-degrees of n_1 nodes and out-degrees of n_2 nodes, the generic sensitivity of publishing the graph is

$$S_{\mathcal{Q}}(G) = \min\{2\min\{n_1, n_2\} + 1, 2|V|\}.$$

Proof. To be added... $\square$

6. CONSISTENCY AND NOISE

After random noise is inserted into C_{base}, although the amount of noise is calibrated in such a way that adversary's background knowledge does not help in recovering the original data, it is not guaranteed that the noisy privacy-preserved table is consistent with the background knowledge. In this section, we present techniques to enforce consistency in generic differentially private tables and graphs. As by-products of consistency, it can be also shown that, in the consistency-enforcing tables and graphs, noise is even reduced.

6.1 Enforcing Consistency in Tables

Recall $\mathsf{c}(b)$ is the real count measure of a cell b, which is known for each cell b in the background knowledge $\mathcal{Q}$. $\tilde{\mathsf{c}}(a)$ is the noisy measure in the base cuboid obtained by calibrating noise according to generic sensitivity. The goal of consistency enforcement is to compute consistent count measure $\hat{\mathsf{c}}(a)$'s from $\tilde{\mathsf{c}}(a)$'s (for cells a in the base cuboid) and $\mathsf{c}(b)$'s (for cells b in the background knowledge $\mathcal{Q}$). From Theorem 4.5, the computation of $\hat{\mathsf{c}}(\cdot)$ preserve the same privacy guarantee as $\tilde{\mathsf{c}}(\cdot)$.

For a cell b in $\mathcal{Q}$, let Base(b) be the set of all base cells, each of which aggregates a disjoint subset of rows that are contained in b. Formally, $a \in \text{Base}(b)$ if and only if for any dimension A_i, $b[i] \neq *$ implies $b[i] = a[i]$. For any cuboid $C \in \mathcal{Q}$ to be released exactly, the *consistency constraints* we want to enforce for measure $\hat{\mathsf{c}}(\cdot)$ is:

$$\sum_{a \in \text{Base}(b)} \hat{\mathsf{c}}(a) = \mathsf{c}(b), \quad \forall \text{ cells } b \in C. \tag{1}$$

Suppose the background knowledge $\mathcal{Q} = \{C\}$ and let Base be the set of all base cells, we enforce the consistency by solving the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{a \in \text{Base}} (\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a))^2 \\ \text{s.t.} \quad & \sum_{a \in \text{Base}(b)} \hat{\mathsf{c}}(a) = \mathsf{c}(b), \quad \forall \text{ cell } b \in C. \end{aligned} \tag{2}$$

Similarly, suppose the background knowledge $\mathcal{Q} = \{C_1, C_2\}$, we enforce the consistency by solving the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{a \in \text{Base}} (\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a))^2 \\ \text{s.t.} \quad & \sum_{a \in \text{Base}(b)} \hat{\mathsf{c}}(a) = \mathsf{c}(b), \quad \forall \text{ cell } b \in C_1. \\ & \sum_{a \in \text{Base}(b)} \hat{\mathsf{c}}(a) = \mathsf{c}(b), \quad \forall \text{ cell } b \in C_2. \end{aligned} \tag{3}$$

In the following part, we will introduce how to find the optimal solutions Problems (2)-(3) in linear time (linear to the number of all possible base cells).

Enforcing Consistency for One Exact Cuboid

The basic idea of solving Problem (2) is to apply the method of Lagrange multipliers. Create a Lagrange multiplier λ_b for each cell constraint in (2), Problem (2) is equivalent to minimizing:

$$\begin{aligned} f(\hat{\mathsf{c}}) = & \sum_{a \in \text{Base}} (\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a))^2 \\ & + \sum_{b \in C} \lambda_b \left(\mathsf{c}(b) - \sum_{a \in \text{Base}(b)} \hat{\mathsf{c}}(a) \right). \end{aligned} \tag{4}$$

Note that $\tilde{\mathsf{c}}(\cdot)$ and $\mathsf{c}(\cdot)$ are constants. So $f(\hat{\mathsf{c}})$ is minimized when $\partial f / \partial \hat{\mathsf{c}} = 0$ in (4). For each cell $a \in \text{Base}$,

$$0 = \frac{\partial f(\hat{\mathsf{c}})}{\partial \hat{\mathsf{c}}(a)} = 2\left(\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a)\right) - \lambda_{a[C]}. \tag{5}$$

Here, for a cell a and a cuboid C, let $a[C]$ be the projection of a on the cuboid C, i.e., the cell $a[C] = b \in C$ s.t. $b[i] = a[i]$ for each dimension $A_i \in [C]$ and $b[i] = *$ for each $A_j \notin [C]$. Obviously, for $a \in \text{Base}$ and $b \in C$, we have $a \in \text{Base}(b) \Leftrightarrow b = a[C]$.

For each cuboid C, let $\deg(C) = \prod_{A_i \notin [C]} |A_i|$ be the number of base cells that aggregate into a cell in C.

Now for each $b \in C$, sum up (5) for all $a \in \text{Base}(b)$:

$$\begin{aligned} & 0 = \sum_{a \in \text{Base}(b)} \left(2\left(\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a)\right) - \lambda_{a[C]} \right) \\ \Leftrightarrow \quad & 0 = 2\left(\mathsf{c}(b) - \sum_{a \in \text{Base}(b)} \tilde{\mathsf{c}}(a) \right) - \deg(C)\lambda_b \\ \Leftrightarrow \quad & \lambda_b = \frac{2}{\deg(C)} \left(\mathsf{c}(b) - \sum_{a \in \text{Base}(b)} \tilde{\mathsf{c}}(a) \right). \end{aligned} \tag{6}$$

Also, for $a \in \text{Base}(b)$, rewrite (5) as:

$$\hat{\mathsf{c}}(a) = \frac{\lambda_b}{2} + \tilde{\mathsf{c}}(a). \tag{7}$$

From (6) and (7), here is a simple linear-time algorithm to solve Problem (2) for the optimal solution $\hat{\mathsf{c}}$.

```
91: For each cell $b \in C$, compute $\lambda_b$ as in (6);
92: For each cell $a \in \text{Base}$, compute $\hat{\mathsf{c}}(a)$ as in (7).
```

Enforcing Consistency for Two Exact Cuboids

The idea of solving Problem (3) is similar to the above, but much more involved. Introduce two sets of Lagrange multipliers λ and γ, one for cells in C_1 and the other for cells in C_2. Problem (3) is equivalent to minimizing:

$$\begin{aligned} f(\hat{\mathsf{c}}) = & \sum_{a \in \text{Base}} (\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a))^2 \\ & + \sum_{b \in C_1} \lambda_b \left(\mathsf{c}(b) - \sum_{a \in \text{Base}(b)} \hat{\mathsf{c}}(a) \right) \\ & + \sum_{b \in C_2} \gamma_b \left(\mathsf{c}(b) - \sum_{a \in \text{Base}(b)} \hat{\mathsf{c}}(a) \right). \end{aligned} \tag{8}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} = \left[\begin{array}{ccccc|ccccc}
d_1 & 0 & \cdots & 0 & 0 & e & e & \cdots & e & e \\
0 & d_1 & \cdots & 0 & 0 & e & e & \cdots & e & e \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & d_1 & 0 & e & e & \cdots & e & e \\
0 & 0 & \cdots & 0 & d_1 & e & e & \cdots & e & e \\ \hline
e & e & \cdots & e & e & d_2 & 0 & \cdots & 0 & 0 \\
e & e & \cdots & e & e & 0 & d_2 & \cdots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
e & e & \cdots & e & e & 0 & 0 & \cdots & d_2 & 0 \\
e & e & \cdots & e & e & 0 & 0 & \cdots & 0 & d_2
\end{array}\right]$$

$d_1 = \deg(C_1)$
$d_2 = \deg(C_2)$
$e = \deg(C_1 \vee C_2)$

$n_1 = \text{size}(C_1)$: number of cells in C_1
$n_2 = \text{size}(C_2)$: number of cells in C_2

$\mathbf{A}_{11}$: $n_1 \times n_1$ matrix
$\mathbf{A}_{22}$: $n_1 \times n_1$ matrix
$\mathbf{A}_{12}$: $n_1 \times n_2$ matrix
$\mathbf{A}_{21}$: $n_2 \times n_1$ matrix

Figure 5: Matrix A in the linear system (12)

Consider the condition of minimizing $f(\hat{\mathsf{c}})$, $\partial f / \partial \hat{\mathsf{c}} = 0$, in (8). For each cell $a \in \text{Base}$, we have

$$0 = \frac{\partial f(\hat{\mathsf{c}})}{\partial \hat{\mathsf{c}}(a)} = 2\left(\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a)\right) - \lambda_{a[C_1]} - \gamma_{a[C_2]}. \tag{9}$$

For each $b_1 \in C_1$, let's sum up (9) for all $a \in \text{Base}(b_1)$:

$$\begin{aligned}
& 0 = \sum_{a \in \text{Base}(b)} \big(2\left(\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a)\right) - \lambda_{a[C_1]} - \gamma_{a[C_2]}\big) \\
\Leftrightarrow \quad & 0 = 2\left(\mathsf{c}(b_1) - \sum_{a \in \text{Base}(b_1)} \tilde{\mathsf{c}}(a)\right) \\
& \qquad - \deg(C_1)\lambda_{b_1} - \sum_{b_2 \in C_2} \deg(C_1 \vee C_2)\gamma_{b_2} \\
\Leftrightarrow \quad & \deg(C_1)\lambda_{b_1} + \sum_{b_2 \in C_2} \deg(C_1 \vee C_2)\gamma_{b_2} \\
& = 2\left(\mathsf{c}(b_1) - \sum_{a \in \text{Base}(b_1)} \tilde{\mathsf{c}}(a)\right) = \mathbf{y}_{b_1}. \qquad (10)
\end{aligned}$$

Similarly, for each $b_2 \in C_2$, summing up (9) for all $a \in \text{Base}(b_2)$, we have:

$$\begin{aligned}
& \sum_{b_1 \in C_1} \deg(C_1 \vee C_2)\lambda_{b_1} + \deg(C_2)\gamma_{b_2} \\
& = 2\left(\mathsf{c}(b_2) - \sum_{a \in \text{Base}(b_2)} \tilde{\mathsf{c}}(a)\right) = \mathbf{y}_{b_2}. \qquad (11)
\end{aligned}$$

Let $\lambda_\bullet$ be a vector with each λ_{b_1} for each $b_1 \in C_1$ as an entry, and $\gamma_\bullet$ be a vector with each γ_{b_2} for each $b_2 \in C_2$ as an entry. (10) and (11) form a system of linear equations with variables $\mathbf{x} = [\lambda_\bullet \ \gamma_\bullet]^\top$:

$$\mathbf{A}\mathbf{x} = \mathbf{y}, \tag{12}$$

where $\mathbf{y}$ is a vector with $\text{size}(C_1) + \text{size}(C_2)$ entries each of which corresponds to a cell in C_1 or C_2, as defined in (10)-(11), and $\mathbf{A}$ is a square matrix with $\text{size}(C_1) + \text{size}(C_2)$ rows/columns, as shown in Figure 5.

With some algebra, (12) can be solved in linear time and thus Problem (3) can be solved in linear time.

Consistency and Noise Reduction

Indeed, feasible solutions to Problems (2)-(3) are consistent to the background knowledge. We will show that we are able to find the optimal solution w.r.t. the L_2-norm objective function.

THEOREM 6.1. *In Problems* (2)-(3)*, the optimal solution* $\hat{\mathsf{c}}(\cdot)$ *can be found in linear time. The measure* $\hat{\mathsf{c}}(\cdot)$ *is consistent to the background knowledge* $\mathcal{Q}$*.*

Moreover, for each base cell $a \in \text{Base}$*,* $\hat{\mathsf{c}}(a)$ *is an unbiased estimator of* $\mathsf{c}(a)$*:* $\mathrm{E}\left[\hat{\mathsf{c}}(a)\right] = \mathsf{c}(a)$*, and the noise is even reduced:* $\mathrm{Var}\left[\hat{\mathsf{c}}(a)\right] < \mathrm{Var}\left[\tilde{\mathsf{c}}(a)\right]$.

PROOF. Too be added... $\Box$

6.2 Enforcing Consistency in Graphs

7. EXPERIMENTS

8. RELATED WORK

9. CONCLUSIONS

This is the conclusion...

Acknowledgments

This is the acknowledgment...

10. REFERENCES

[1] A. Dobra. Markov bases for decomposable graphical models. *Bernoulli*, 9(6):1–16, 2003.
[2] C. Dwork. Differential privacy. In *ICALP (2)*, pages 1–12, 2006.
[3] C. Dwork. Differential privacy: A survey of results. In *TAMC*, pages 1–19, 2008.
[4] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.
[5] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total. In *ICDE*, pages 152–159, 1996.

[6] D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *SIGMOD*, pages 193–204, 2011.

[7] D. Kifer and A. Machanavajjhala. A rigorous and customizable framework for privacy. In *PODS*, pages 77–88, 2012.

[8] F. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*, pages 19–30, 2009.

[9] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103, 2007.

APPENDIX

This the appendix...