Technical Report

TR-2010-003

Non-interactive data release for distributive queries with differential privacy

by

Yonghui Xiao, Li Xiong, Chun Yuan

Mathematics and Computer Science

EMORY UNIVERSITY

Non-interactive data release for distributive queries with differential privacy

ABSTRACT

Differential privacy is emerging as a strong notion for protecting individual privacy in privacy preserving data analysis or publishing. While it has been proven that it is possible to non-interactively release a database that are useful for all queries satisfying certain constraints with differential privacy, there is still a lack of general and efficient methods for achieving the data release. We propose a novel and efficient method for non-interactive data release for distributive queries with differential privacy. We examine queries with unlimited and limited output domain respectively and propose different differential privacy mechanisms which simultaneously guarantees differential privacy and the usefulness of released data. We also instantiate the general method for different queries including interval queries, parity queries and max queries. In addition to providing formal proofs of the differential privacy and usefulness guarantees of the released data using our method, we also present a set of experimental results and demonstrate the actual feasibility and performance of our method for different queries.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Algorithm, Security

Keywords

differential privacy, non-interactive, distributive query

1. INTRODUCTION

As current information technology enables many organizations to collect, store, and use various types of information about individuals, privacy protection becomes an increasingly important issue. Government and organizations increasingly recognize the critical value in sharing such information while preserving the privacy of individuals.

Privacy preserving data analysis and data publishing [7, 14] has received considerable attention in recent years as a promising approach for sharing useful information while preserving data privacy. There are two models for privacy protection [7]: interactive model and non-interactive model. In the interactive model, a trusted *curator* (e.g. hospital) collects data from *record owners* (e.g. patients) and provides an access mechanism for *data users* (e.g. public health researchers) for querying or analysis purposes. The answer returned from the access mechanism usually was modified by the mechanism to protect privacy. In the non-interactive model, the curator publishes a "sanitized" version of the data by using certain techniques, simultaneously providing utility for data user and privacy protection for data owners.

It has been shown that absolute privacy protection is impossible for privacy preserving data analysis due to the presence of background knowledge [5]. Most literature [14] following the seminal work on k-anonymity [22, 24] and l-diversity [19] adopts relaxed privacy notions for the non-interactive model by considering specific attacks and assuming the attacker has limited background knowledge. Differential privacy [10, 6, 8, 25] is emerging as a strong notion for guaranteeing privacy with arbitrary background knowledge. Initially proposed in the interactive model, the key notion of differential privacy is that even if an adversary happens to know all the records except the record of a victim, she still cannot predict it with high probability in queries sublinear in the number of database rows.

Many meaningful results have been archived for interactive model that achieves differential privacy [10, 6, 8, 1]. However, to guarantee differential privacy, the total number of queries must be sublinear in the number of database rows, called SuLQ(Sub-Linear Queries). With this requirement, an interactive mechanism must be shut down after the number of queries reaches the upper bound, which limits the development of interactive model for the practical need of data publishing or exchange. In contrast to the above work focused on interactive mechanisms, Blum et al. [3] recently proved the possibility of non-interactively releasing data with differential privacy for queries from concept classes with polynomial VC-dimension, such as predicate queries [1].

[1]Predicate query is of the form that "what is the fraction of data indexed by the predicate"

Although Blum et al. [3] has given an inefficient general algorithm for releasing data for learning a concept whose hypothesis is measured by predicate queries, to our knowledge, there is no general efficient method for releasing data with differential privacy. It remains an open question that for what kind of queries can we release the data efficiently. Machanavajjhala et al.[20] generate synthetic data that statistically mimic the original data without demonstration of the utility of the synthetic data. Feldman et al.[11] give an algorithm used the notion "private coreset" to release the certain query: k-median, k-mean and k-center.

In this paper, we study the problem of non-interactive privacy preserving data release that achieves differential privacy. We propose a general and efficient method for releasing data to support *distributive* statistical queries. Introduced in the data warehousing context [18], a distributive measure is a kind of measure that can be computed in a distributed manner. Suppose the data are divided into n partitions (at distributed sites in the original context). If the result derived by applying the query on each of the n partitions and then aggregating the sub-results is the same as that derived by applying the query on all data without partitioning, the query is called a distributive query. For example, sum() can be computed either by summing up all the data values or by first partitioning the data then summing up the sums of each partition. Motivated by this property, the key idea of our general method is to release the data using a partitioning approach. First we divide the data into partitions and then release the data for each partition with differential privacy. By partitioning the data into smallest cells, we could utilize or design an interactive mechanism that answer a query with differential privacy and release data that are consistent with the result achieved by the mechanism. Given the distributive property of the query, we can guarantee the overall data are differentially private.

The remaining question is how to guarantee the released data is useful to the original query. To measure the utility of released data, we use the criterion of (ϵ, δ)-usefulness [3], which guarantees that the answer to the queries are "almost" correct with probability $1-\delta$. In our partitioning approach, the answer for the original query is aggregated from the results of each individual partitions. By examining different queries, we propose different instances of our general method to guarantee the (ϵ, δ)-usefulness while satisfying differential privacy. For queries such as sum() and count(), the domain of the output is not limited in certain range as long as the data size is arbitrary. For these queries, we prove that the result is (ϵ, δ)-useful by utilizing a Laplace noise approach as the interactive mechanism [10, 6, 1] for releasing individual partitions. On the other hand, for queries such as max() and min(), the domain of the output must be the same as the domain of input data. For such queries, using a Laplace noise approach [10, 6, 1] would severely affect the data utility as the noise could be so big that the output result could go beyond the domain. we proposed a method based on the exponential mechanism [21] for such queries and prove its privacy and utility guarantee.

1.1 Contributions

1. General releasing method for distributive queries. While [3] proved the possibility of non-interactively releasing a database that is useful for all queries over discretized domain from any concept class with polynomial VC-dimension, to our knowledge, there was no general efficient method for data releasing. Our first contribution is a general and efficient method that can be used to release a database for distributive queries so that it achieves differential privacy. Motivated by the distributive property, our method uses a novel partitioning idea and releases the data by partitioning the data into cells and releasing data for each cell utilizing interactive mechanisms to achieve differential privacy.

2. A new algorithm for distributive queries with limited output domain. We show how the general method can be instantiated for different queries in order to achieve a guaranteed data utility. The few works on non-interactive mechanisms for achieving differential privacy focus on predicate queries [3], k-median, k-mean and k-center queries [11], whose output is not restricted on limited domain. We propose a mechanism as an instance of our general method for distributive queries whose output is on limited domain such as max queries.

3. Formal proofs and experimental results of the method. Most works on differential privacy remain theoretical. In this paper, in addition to formal proofs on the differential privacy and (ϵ, δ)-usefulness guarantees of our methods, we also present a set of experimental evaluations and show the actual performance of our method for interval query, parity query, and max query respectively. We compare our algorithm with the algorithm proposed for interval queries in [3], showing the efficiency and extendability of our algorithm. For parity queries, we also learn the parity vector from the released data to demonstrate the usefulness of the release data in terms of learnability in addition to the (ϵ, δ)-usefulness. For max queries with limited output domain, we show that the utility of the data is related to the data distribution and data size.

2. RELATED WORKS

Privacy preserving data analysis or publishing has received considerable attention in recent years. We refer readers to [7, 14] for a mostly complete and up-to-date survey. We briefly review here the most relevant work to our paper and discuss how our work differs from them.

There has been a series of studies on interactive privacy preserving data analysis based on the notion of differential privacy [10, 6, 8]. The notion of differential privacy provides strong privacy guarantee and is generalized into both interactive and non-interactive settings. There are recently a few works focused on non-interactive data release that achieves differential privacy [10, 3, 11]. Blum et al. [3] proved the possibility of non-interactive data release satisfying differential privacy for queries with polynomial VC-dimension, such as predicate queries. However, the result remains theoretical and the general algorithm is inefficient for the complexity and required data size. Feldman et al. [11] proposed the notion "private coreset" to release data for certain queries: k-median, k-mean, k-center. Machanavajjhala et al. [20] show a method to release synthetic data of the commuting patterns of the population, which statistically mimic the original data without formal demonstration of the utility of the synthetic data.

Our work complements and advances the above works in that we propose a novel and efficient general method as well as specific algorithms for releasing data for distributive queries including those with an output on limited domain such as max queries which are not previously studied.

A primary approach proposed for achieving differential privacy is to add Laplace noise [10, 8, 6]. McSherry and Talwar[21] give a new way to implement differential privacy based on probability, called "exponential mechanism", which is used in [3] to demonstrate the inefficient general algorithm. Our work utilizes the Laplace noise approach in the non-interactive approach and also proposes a modified "exponential mechanism" for releasing data supporting queries on limited output range.

The problem of parity function has been well studied as a learning problem. [13, 15] seeks the hidden vector v by solve linear equations. [2] has proved that parity can be learned with random classification noise. [16] has proposed a private parity learning algorithm directly from the original database rather than the released data. [10] uses the parity function to separate the interactive and non-interactive model of data release approach. The method in [10] to guarantee differential privacy is a randomized operator $Z : D \rightarrow \{0,1\}^*$. But the method perturbed every attribute of the data, which is usually not necessary.

3. PRELIMINARIES AND DEFINITIONS

Given an original database D, we use $\mathcal{A}(D)$ to denote an mechanism to access the database D. Any query $Q(D)$, gets a returned result $\mathcal{A}_Q(D)$ from the interactive query answering mechanism. In non-interactive mechanism, for any query $Q(D)$, our goal is to release a database $\hat{D}$ to answer the query, which satisfies differential privacy and (ϵ, δ)-usefulness for the query $Q(D)$.

In this section, we formally introduce the definitions on differential privacy as well as existing mechanisms for achieving differential privacy, the definition of (ϵ, δ)-usefulness, and the notion of data cube to facilitate our discussions.

3.1 Differential Privacy

DEFINITION 3.1 (α-DIFFERENTIAL PRIVACY[6]). *In interactive model, an access mechanism $\mathcal{A}$ satisfies α-differential privacy if for any neighboring database D_1 and D_2, for any query function Q, $r \subseteq Range(Q)$, $\mathcal{A}_Q(D)$ is the mechanism to return an answer to query $Q(D)$,*

$$Pr[\mathcal{A}_Q(D_1) = r] \leq \epsilon^{\alpha} Pr[\mathcal{A}_Q(D_2) = r]$$

In non-interactive model, a data release mechanism $\mathcal{A}$ satisfies α-differential privacy if for all neighboring database D_1 and D_2, all released output $\hat{D}$,

$$Pr[\mathcal{A}(D_1) = \hat{D}] \leq \epsilon^{\alpha} Pr[\mathcal{A}(D_2) = \hat{D}]$$

To achieve differential privacy, there are generally two ways: fixed-data based and probability based method.

3.1.1 Fixed-data perturbation

For fixed-data perturbation, we need to add random noise of Laplace distribution to the true answer of the query Q,

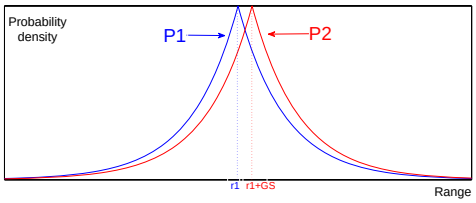

Figure 1: Probability curve

$\mathcal{A}(x) = Q(x) + Y$, where Y is the Laplace noise. Global sensitivity is the parameter of Y.

DEFINITION 3.2 (GLOBAL SENSITIVITY). *For any neighboring database D_1 and D_2, global sensitivity of a query Q is the maximum difference between the query results of D_1 and D_2,*

$$GS_Q = max|Q(D_1) - Q(D_2)|$$

If we want to achieve α-differential privacy, we just draw Y from $Lap(GS_Q/\alpha)$[10].

3.1.2 Probability perturbation

A query answering mechanism is to map a set of input from a domain $\mathcal{D}$ to some output in a range $\mathcal{R}$. By designing an evaluation function $q : \mathcal{D}^n \times \mathcal{R} \rightarrow \mathbb{R}$ to assign a value to any pair (d, r), exponential mechanism is such a map that output $r \in \mathcal{R}$ with probability proportional to $exp(\frac{\alpha q(d,r)}{2\Delta q})$ which has been proven that gives α-differential privacy[21], where Δq is the largest difference of q between two neighboring input which differs only one record.

The two methods are the same from the view of probability. For two neighboring database D_1 and D_2, $\mathcal{A}(D_1) = Q(D_1) + Y$, where Y is the Laplace noise. In Figure 1, the curve P_1 is the probability density of $\mathcal{A}(D_1)$, where $r_1 = Q(D_1)$. The x-axis is the output range of query and y-axis is the probability density of the range value. Because GS is the max difference between D_1 and D_2, $r_1 - GS \leq Q(D_2) \leq r_1 + GS$. Therefore, the probability curve of $\mathcal{A}(D_2)$ can move left or right from r_1 at most GS, showing as P_2 in Figure 1. For any given r, $r \in \mathcal{R}$, it's easy to prove $1/e^{\epsilon} \leq P_1(r)/P_2(r) \leq e^{\epsilon}$. As the data size grows, the probability curve moves right gradually, but the shape of the curve remains the same. It's a characteristic of fixed-data perturbation. For the probability perturbation, the probability curve may vary according to the data.

Usually for output range $\mathcal{R}$ restricted on limited discretized domain, it's easy to implement probability perturbation; For output range $\mathcal{R}$ on unlimited domain, it's easy to implement fixed-data perturbation.

3.2 Data cube

We use the notation "data cube" to present the data space $\mathcal{D}^n$. For example, if the database has N dimensions, it is a N-dimensional cube. We denote the data cube by Ω, a query $Q : \Omega \rightarrow \mathcal{R}$ is to map the data in Ω to output range $\mathcal{R}$. All the records in the database are points in the data

cube. We use the notation "partition" to present any sub-cube in the data cube. We denote any sub-cube that cannot be subdivided in any dimensions by "cell", meaning it's the "smallest" sub-cube. We denote the number of cells by β.

3.3 (ϵ, δ)-usefulness

To guarantee utility of released data, we consider the releasing mechanism is useful if it's (ϵ, δ)-useful.

DEFINITION 3.3 ((ϵ, δ)-USEFULNESS[3]). *A database mechanism $\mathcal{A}$ is (ϵ, δ)-useful for queries in class C if with probability $1-\delta$, for every $Q \in C$, and every database D, $\mathcal{A}(D) = \hat{D}$, $|Q(\hat{D}) - Q(D)| \leq \epsilon$.*

4. GENERAL RELEASING METHOD

An (inefficient) data releasing mechanism is given in [3] by using the exponential mechanism of [21] to output a $\hat{D}$ which is (ϵ, δ)-useful. But the mechanism is not practically feasible for the complexity and data size. Another mechanism is given for k-median, k-mean and k-center queries[11]. We focus on distributive query and propose a simple and efficient general releasing mechanism protecting α-differential privacy and maintaining (ϵ, δ)-usefulness.

In interactive model, users can ask queries to the access mechanism. As long as the total number of queries are sublinear in the number of database rows, mechanism of differential privacy make sure that adversary can not have higher confidence to know the sensitive data than prior probability. Equally, if we release the data that are consistent with the query results as if they were returned by a mechanism of differential privacy, it also protects privacy of data owners, which is the intuition of our method.

4.1 General Releasing method

The initial model of differential privacy is to answer statistical query which is an aggregate information indexed by some query parameters. Informally speaking, statistical query is like a crowd in which individuals are hidden. We cannot let users query information of certain record because that will definitely violate privacy if we want to maintain the preciseness of query answer. Therefore, to release data, we still only can release statistical query result instead of precise record information. In the data cube, the query answering process is to move some points from the original partitions to other partitions and return the perturbed information. So why don't we directly ask the cell information of the database?

DEFINITION 4.1 (DISTRIBUTIVE QUERY[18]). *A distributive query is a function that can be computed for a given data set by partitioning the data into small subsets, computing the function of each subset, and then merging the results in order to arrive at the function's value for the original(entire) data set.*

For example, sum() can be computed either by summing up all the data values or by first partitioning the data then summing up the sums of each partition. Hence, sum() is a distributive query. We give the general releasing algorithm as follows:

Steps:
1. Divide the data into partitions/cells according to some dimensions or the output range of $Q(x)$.
2. Query every partition/cell with interactive mechanism of α-differential privacy.

   For queries whose output range is unlimited, we achieve differential privacy by fixed-data perturbation;

   For queries whose output range is limited on certain domain, we achieve differential privacy by probability perturbation.

3. Release the results of the query or release the data consistent of the query results.

Note that it doesn't matter we release the query results or the data consistent of the query results as long as we make sure that the released data is (ϵ, δ)-useful.

We separate the approach to achieve differential privacy by the output range of query. By the term "output range on unlimited domain", we mean that by adding any variable from Laplace distribution, the returned result of the query could be possibly useful; By the term "output range on limited domain", we mean that by adding some variable from Laplace distribution, the returned result of the query could be meaningless, for example, people'age could not be possibly larger than 150.

4.1.1 For output range on unlimited domain

For queries whose output range is unlimited, we achieve differential privacy by fixed-data perturbation, which add random noise of Laplace distribution to the true answer of the query Q, $\mathcal{A}_Q(x) = Q(x) + Y$, where Y is the Laplace noise $\text{Lap}(GS/\alpha)$. Because we divide the data into β cells, the computation time is $O(\beta)$.

THEOREM 4.1. *By using fixed-data perturbation, the algorithm above achieves α-differential privacy for distributive query with output range on unlimited domain .*

PROOF. Because for every cell, the query is answered by the mechanism of α-differential privacy, the releasing of query results is α-differentially private. □

THEOREM 4.2. *For distributive query $Q(D) = \sum Q(D_i)$, the released $\hat{D}$ of algorithm above maintain (ϵ, δ)-useful if $GS \leq \frac{\alpha\epsilon}{\beta ln(\beta/\delta)}$, where β is the number of partitioned cells in data cube.*

PROOF. By interactive mechanism of differential privacy, the returned answer $\mathcal{A}_Q(D) = Q(D)+Y$, where $Q(D)$ is the true answer of the query and Y is the Laplace noise Lap(b) where b=GS/α. we use D_i to present the data in the cells, then the returned answer of $\hat{D}$ is

$$Q(\hat{D}) = \sum_{i=1}^{\beta}(Q(D_i)+Y_i) = \sum_{i=1}^{\beta} Q(D_i)+\sum_{i=1}^{\beta} Y_i = Q(D)+\sum_{i=1}^{\beta} Y_i$$

$$|Q(D) - Q(\hat{D})| = |Q(D) - (Q(D) + \sum_{i=1}^{\beta} Y_i)| \leq \sum_{i=1}^{\beta} |Y_i|$$

LEMMA 4.1. *If Y_i is the random variables i.i.d from Lap(b) with mean 0, then*

$$Pr[\sum_{i=1}^{\beta} |Y_i| \leq \epsilon] \geq 1 - \beta \cdot exp(-\frac{\epsilon}{\beta b})$$

With Lemma 4.1, we have

$$Pr[|Q(x, D) - Q(x, \hat{D})| \leq \epsilon] \geq 1 - \beta \cdot exp(-\frac{\epsilon}{\beta b})$$

If $\beta \cdot exp(-\frac{\epsilon}{\beta b}) \leq \delta$, then we can get

$$Pr[|Q(x, D) - Q(x, \hat{D})| \leq \epsilon] \geq 1 - \delta$$

So, $\beta \cdot exp(-\frac{\epsilon}{\beta b}) \leq \delta$, b=GS/$\alpha$, we have

$$GS \leq \frac{\alpha\epsilon}{\beta ln(\beta/\delta)}$$

□

Therefore, by control the upper bound of GS, which is often related to data size n, we can guarantee the (ϵ, δ)-usefulness of released data.

PROOF OF LEMMA 4.1. We assume each $|Y_i| \leq \epsilon_1$ where $\epsilon_1 = \epsilon/\beta$. Otherwise we call $|Y_i| > \epsilon_1$ a FAILURE. If no FAILURE happens,we have

$$\sum_{i=1}^{\beta} |Y_i| \leq \beta \cdot \epsilon_1 = \epsilon$$

If a FAILURE happens, then we have $|Lap(b)| > \epsilon_1$, which means

$$\text{Pr[a FAILURE]}=2\int_{\epsilon_1}^{\infty} \frac{1}{2b} exp(-\frac{x}{b})=e^{-\epsilon_1/b}$$

For each Y_i, Pr[no FAILURE happens]=1-Pr[FAILURE happens] and each Y_i is i.i.d distributed, we have

$$Pr[\sum_{i=1}^{\beta} |Y_i| \leq \epsilon] \geq (1 - e^{-\epsilon_1/b})^{\beta}$$

Let $F(x) = (1-x)^{\beta} + \beta x - 1$, then $F(0) = 0$. $F'(x) = -\beta(1-x)^{\beta-1} + \beta = \beta(1 - (1-x)^{\beta-1}) > 0$ when $0 < x < 1$. Note that $0 < e^{-\epsilon_1/b} < 1$, so $F(e^{-\epsilon_1/b}) > 0$. We get $(1 - e^{-\epsilon_1/b})^{\beta} > 1 - \beta \cdot e^{-\epsilon_1/b}$. Because $\epsilon_1 = \epsilon/\beta$,

$$Pr[\sum_{i=1}^{\beta} |Y_i| \leq \epsilon] \geq 1 - \beta \cdot exp(-\frac{\epsilon}{\beta b})$$

□

4.1.2 For output range on limited domain

For queries whose output range is on limited domain, such as max() and min(), we use the probability perturbation to achieve differential privacy. To guarantee differential privacy, we should give a dynamic result that many different answers could possibly be given according to some probability with the understanding that the more the answer is close to the true answer, the more the probability is. We use a little modified "exponential mechanism" to achieve the algorithm. Let the evaluation function q be the same as query $Q(x)$ where $x \in \mathcal{D}^n$ and $Q(x) \in \mathcal{R}$. Assume the difference of two domain value is 1. $Q(x)$ has β different possible results in the range $\mathcal{R}$. We denote the output probability of neighboring database D_1 and D_2 by $Pr[D_1]$ and $Pr[D_2]$ respectively. When a point x_i is added to the database D_1, D_1 becomes D_2. With the understanding of the query Q, we denote the output range that may have positive effect to $Q(x_i)$ by r^+; denote the output range that may have negative effect to $Q(x_i)$ by r^-; if some ranges have no effect to $Q(x_i)$, we do nothing on the probability of the range. By iterating every point in the database, we increase the probability of r^+ and decrease the probability of r^- within the proportion of e^{α}. Then the probability of output range is changed gradually. The computation time is $O(n)$ where n is the data size.

Steps for releasing queries whose output range on limited domain:
1. uniformly generate a probability distribution with β cells, meaning the probability of output value r is $1/\beta$, $Pr[Q(D) = r] = 1/\beta$.
2. For each data x_i in the database

    Compute the probability mass of the range r^+, denote by PM_2;

    Compute the probability mass of the range r^-, denote by PM_1;

    Let $t_2 = PM_2 * (\epsilon^{\alpha} - 1)$, $t_1 = PM_1 * (1 - 1/\epsilon^{\alpha})$;

    if $t_2 < t_1$

    add total probability t_2 to the range r^+ with increment of each cell proportional to their distribution;

    minus total probability t_2 to the range r^- with decrement of each cell proportional to their distribution;

    else

    add total probability t_1 to the range r^+ with increment of each cell proportional to their distribution;

    minus total probability t_1 to the range r^- with decrement of each cell proportional to their distribution;

    end

3. draw a data as the result of query Q according to probability of output range $Pr[r]$.

THEOREM 4.3. *The algorithm above is α-differentially private.*

PROOF. The algorithm processes data one by one to change the output probability. Therefore, if we can prove from any neighboring database D_1 to D_2, the algorithm is differentially private, according to the definition of differential privacy, the whole algorithm satisfies differential privacy.

By adding the differing point x_i to D_1, D_1 becomes D_2. The probability mass in the range r^+ will increase; the probability mass in the range r^- will decrease. If $t_2 < t_1$, the probability mass in the range r^+ increase by t_2, then

$$Pr[Q(D_2) \in r^+] = PM_2 + t_2 = PM_2 * e^\alpha$$
$$\frac{Pr[Q(D_2) \in r^+]}{Pr[Q(D_1) \in r^+]} = PM_2 * e^\alpha / PM_2 \leq e^\alpha$$

If $t_2 \geq t_1$, the probability mass in the range r^- will decrease by t_1, then

$$Pr[Q(D_2) \in r^-] = PM_1 - t_1 = PM_1/e^\alpha$$
$$\frac{Pr[Q(D_2) \in r^-]}{Pr[Q(D_1) \in r^-]} = \frac{PM_1/e^\alpha}{PM_1} \geq 1/e^\alpha$$

So we get

$$1/e^\epsilon \leq \frac{Pr[Q(D_2) \in \mathcal{R}]}{Pr[Q(D_1) \in \mathcal{R}]} \leq e^\epsilon$$

It's differentially private. □

Generally, the usefulness of the algorithm depends on the query itself, probability density function and data size. In the extreme case, if the data only has 1 record, then the output probability must be the same as the original distribution to protect the only data, which is totally useless for data user. If we assume r^+ is the range $[Q(x_i)-\epsilon, Q(x_i)+\epsilon]$, r^- is the all the ranges except r^+, the data is uniformly distributed, the following conclusion exists.

LEMMA 4.2. *For uniform distribution, the released $\hat{D}$ of the algorithm above is $(\epsilon, \frac{e^\alpha}{e^\alpha+1})$-useful if $n \geq \frac{\beta}{\alpha\epsilon} ln \frac{\beta}{\epsilon(e^\alpha+1)}$ where $\delta \geq \frac{1}{e^\alpha+1}$.*

PROOF. Because of uniform distribution, there will be $\frac{\epsilon}{\beta}n$ data in the range $[Q(D) - \epsilon/2, Q(D) + \epsilon/2]$. By the bound of n, we know that at least $\frac{1}{\alpha} ln \frac{\beta}{\epsilon(e^\alpha+1)}$ data in the range $[Q(D)-\epsilon/2, Q(D)+\epsilon/2]$. At the beginning of the algorithm, the probability at the range $[Q(D)-\epsilon/2, Q(D)+\epsilon/2]$ will be ϵ/β, which means $t_2 < t_1$ in the algorithm. By processing each point of the database, it will multiply the probability by e^α. The probability will become

$$\frac{\epsilon}{\beta}(e^\alpha)^{(\frac{1}{\alpha} ln \frac{\beta}{\epsilon(e^\alpha+1)})} = \frac{1}{e^\alpha+1}$$

For point in the range $[Q(D)-\epsilon/2, Q(D)+\epsilon/2]$, if $Pr[Q(\hat{D}) \in [Q(D) - \epsilon/2,\ Q(D) + \epsilon/2]] \leq \frac{1}{e^\alpha+1}$, then $t_2 \leq t_1$, which is proven as follows.

Let $PM_2 = x$, $PM_1 = 1 - x$. Solve the equation

$$t_1 - t_2 = (1 - x)(1 - 1/e^\alpha) - x(e^\alpha - 1) = 0$$

We get $x = \frac{1}{e^\alpha+1}$. Then $\delta = 1 - \frac{1}{e^\alpha+1} = \frac{e^\alpha}{e^\alpha+1}$. □

THEOREM 4.4. *For uniform distribution, the released $\hat{D}$ is (ϵ, δ)-useful if $n \geq \frac{\beta}{\alpha\epsilon} ln \frac{\beta}{\epsilon\delta(e^\alpha+1)^2}$ where $\delta < \frac{1}{e^\alpha+1}$.*

PROOF. We denote the ranges except $[a, b]$ by $\mathcal{R} - [a, b]$. With Lemma 4.2, we know if $n \geq \frac{\beta}{\alpha\epsilon} ln \frac{\beta}{\epsilon(e^\alpha+1)}$, $\Pr[Q(\hat{D}) \in \mathcal{R} - [Q(D) - \epsilon/2, Q(D) + \epsilon/2]] \leq \frac{1}{e^\alpha+1}$, then $t_1 > t_2$ in the algorithm, so by adding $\frac{1}{\alpha} ln \frac{1}{\delta(e^\alpha+1)}$ points to the range $[Q(D) - \epsilon/2,\ Q(D) + \epsilon/2]$, $\Pr[Q(\hat{D}) \in \mathcal{R} - [Q(D) - \epsilon/2, Q(D) + \epsilon/2]]$ become less than

$$\frac{\frac{1}{e^\alpha+1}}{(e^\alpha)^{\frac{1}{\alpha} ln \frac{1}{\delta(e^\alpha+1)}}} = \delta$$

So $\Pr[Q(\hat{D} \in [Q(D) - \epsilon/2,\ Q(D) + \epsilon/2]]$ become larger than $1 - \delta$ with

$$n \geq \frac{\beta}{\alpha\epsilon} ln \frac{\beta}{\epsilon(e^\alpha+1)} + \frac{\beta}{\epsilon\alpha} ln \frac{1}{\delta(e^\alpha+1)} = \frac{\beta}{\alpha\epsilon} ln \frac{\beta}{\epsilon\delta(e^\alpha+1)^2}$$

□

4.2 Some Remarks about Our Method

4.2.1 Adversary's view

The initial scenario of differential privacy is to assume that even if an adversary happens to know all the records except target individual, she still cannot predict it with high probability in queries sublinear in the number of database rows. It also works for our algorithm. We assume some identifier of individuals is included in the attributes of released data. First, if the identifier are not enough to identify a record, the adversary cannot ascertain which one is the target victim, which is like the k-anonimity[23] mechanism. Second, the query is designed to ask the statistical information of certain cell. If the identifier can be used to identify a record, by the query answering mechanism of α-differential privacy, even if an adversary happens to know all the records in the cell, she cannot predict whether a target individual is in the cell in only one time of query. Similarly, she cannot predict which cell a target victim is in.

4.2.2 Cell correlation

If an attribute only have three values "A, B, C", and an adversary learn from queries that it's not "A" and "B", Could she be sure that the value is "C"? The question means what happens if there is correlation among cells.

THEOREM 4.5. *Composition of differential privacy [9, 17, 16]. If an algorithm $\mathcal{A}$ runs k algorithms $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ where each $\mathcal{A}_i (i = 1, 2, \ldots, k)$ is α_i-differentially private and outputs a function of these results($\mathcal{A}(x) = g(\mathcal{A}_1(x), \mathcal{A}_2(x), \cdots, \mathcal{A}_k(x))$), then $\mathcal{A}$ is $\sum_{i=1}^k \alpha_i$-differentially private.*

The query that the value of the attribute is "A" or "B" each preserves α-differential privacy. With the theorem above, we know that the query result that the value of the attribute is not "A" and "B" preserves 2α-differential privacy. So the adversary still cannot predict the value of the attribute is "C" with good probability by cell correlation.

5. INTERVAL QUERY

Many papers have researched Interval problem, like [3, 4]. However, [4] did not guarantee differential privacy and [3] was not feasible for practical algorithm. Our algorithm can overcome the drawbacks and guarantee the utility of the released data. We first describe the interval query, then give our algorithm and compare the algorithms of [3] with ours. A detailed experiment result is given in section 8.

5.1 Definition

An Interval query on a one dimensional database D is $Q_{[a,b]}(D)$ which asks the total number of points in the interval $[a, b]$. Formally,

DEFINITION 5.1. *An interval query on a one dimensional database D is defined to be:*

$$Q_{[a,b]}(D) = \frac{|\{x \in D : if\ a \le x \le b\ then\ 1\ else\ 0\}|}{|D|}$$

Interval query is similar to histogram query and can be viewed as the sum of several query answers of histograms. For two neighboring database D_1 and D_2, the maximum difference of interval query between D_1 and D_2 is 1, so the Global sensitivity GS_Q is $1/n$ where n is the rows of the database.

5.2 Algorithm

We denote all cells in the range $[a, b]$ by ω, each cell by i, $Q_{[a,b]} = \sum_{i\in\omega} f(i)$.

We use these notations for the algorithm: β: the maximum cells. n: database size. b: the parameter of Laplace noise, Lap(b) where $b = 1/(\alpha n)$. c: the perturbed total number of each cell.

We give the algorithm as follows.

Steps:
1. get the original data distribution ordered by cell.
2. perturb the probability mass in every cell to c with Lap(b).
3. if $c > 0$, output c*n and the cell.

The algorithm return the cells and count number in the cell. Note that it's not necessary to guarantee $|\hat{D}| = n$.

The algorithm can be easily extended to high dimensional database.

5.3 α-differential privacy

THEOREM 5.1. *The algorithm preserves α-differential privacy.*

PROOF. The algorithm is a mechanism $\mathcal{A}$ to output a released database $\hat{D}$. The GS of interval query is $1/n$, therefore, we use the Lap($1/\alpha n$) to perturb the true answer of interval query, which satisfies α-differential privacy, meaning that an adversary cannot tell the difference between two neighboring database D_1 , D_2. When $c > 0$, we have

$$Pr[\mathcal{A}(D_1) = \hat{D}] \le e^{\alpha} Pr[\mathcal{A}(D_2) = \hat{D}];$$

Table 1: Comparison of algorithms

	ours	Algorithm in [3]
Samples	$n \ge \frac{\beta ln(\beta/\delta)}{\alpha\epsilon}$	$n \ge \frac{32b[log(2b)+log(2/\epsilon\delta)]}{\alpha\epsilon^3}$
Distribution	any distribution	uniform distribution
Extendibility to high dimension	easy	hard

when $c \le 0$, the total number in a cell cannot be negative and we cannot output a database with negative total number. We output 0 when $c \le 0$ and demonstrate it also satisfy differential privacy:

$$\frac{Pr[\mathcal{A}(D_1) = 0]}{Pr[\mathcal{A}(D_2) = 0]} = \frac{Pr[t \le 0]}{Pr[t \le 0]} = \frac{\int Pr[t]dt}{\int Pr[t]dt} = \int \frac{Lap(t - Q(D_1))}{Lap(t - Q(D_2))}dt \le \int e^{\alpha}dt < e^{\alpha}$$

where t is the result of $\mathcal{A}(D_1)$ and $\mathcal{A}(D_2)$.

So we have $\frac{Pr[Q(D_1)=0]}{Pr[Q(D_2)=0]} < e^{\epsilon}$ when $c \le 0$. □

5.4 (ϵ, δ)-usefulness

THEOREM 5.2. *The released $\hat{D}$ of the algorithm maintain (ϵ, δ)-useful if $n \ge \frac{\beta ln(\beta/\delta)}{\alpha\epsilon}$.*

PROOF. With Theorem 4.2, we know if $GS \le \frac{\alpha\epsilon}{\beta ln(\beta/\delta)}$, $\hat{D}$ is (ϵ, δ)-useful. $GS = 1/n$. we get $n \ge \frac{\beta ln(\beta/\delta)}{\alpha\epsilon}$ □

5.5 Comparison of algorithms

We compare our algorithm with [3]. First, the samples needed in our algorithm is much less than in [3]. Second, our algorithm can deal with any distribution, while algorithm in [3] can only deal with uniform distribution. With algorithm in [3], in an extreme case that all the points in D has value 0(no other value), we cannot find a sub-interval with probability mass $[\epsilon_1/2 - \epsilon_2, \epsilon_1/2 + \epsilon_2]$ because any sub-interval's probability mass is either "1+noise"(if it contain 0) or "0+noise"(if it don't contain 0). In general cases like gaussian distribution, if the value of ϵ is given, it happens in such a way: suppose the probability mass of an interval is p and the interval cannot be subdivided further more, if $p > \epsilon_1/2 + \epsilon_2$, how can we output the data in the interval? This is the main drawback of algorithm in [3]. Third, our algorithm can be easily extended to high dimensional database, while algorithm in [3] is hard to extend because it's hard to union the cells in high dimensional space. Table 1 shows the comparison result of the algorithms.

6. PARITY FUNCTION

Parity class as a learning problem has been well studied. [13, 15] seeks the hidden vector v by solve linear equations. [2] has proved that parity can be learned with random classification noise. [16] has proposed a private parity learning algorithm directly from the original database rather than the released data. We give our releasing algorithm and prove the released $\hat{D}$ is not only (ϵ, δ)-useful but also learnable. Detailed released and learning result is shown in section 8.

6.1 Definition

Given a non-zero vector $v \in \{0,1\}^{d-1}$, a data set D $\in \{0,1\}^d$, let k be the kth row of the data set and $x_k^{(j)}$ is the jth attribute of x_k. Without loss of generality, we define parity function as the inner product of v and the first $d-1$ attributes of x_i modulo 2 and the result is saved to the dth attribute of x_i, that is

$$g(k,v) = \oplus_{j \le d-1} x_k^{(j)} v^{(j)} = x_k^{(d)},$$

Unlike the interval query, if there exits a parity vector in the data, the distribution is affected because $x^{(d)}$ must conform to the parity pattern. So some cells in the data cube will have no point. Does this give an adversary auxiliary information to approximate an individual's data with more confidence? We say the answer is negative for the following reason. First, in practical case, not all samples are consistent with the parity hypothesis; Second, one cannot predict sensitive data by cell correlation; Third, before learning the parity class, one even have no idea if there exits a parity vector in the data. Even if the parity is learned with high confidence, if we can make sure that adversaries cannot locate the position of target victim in the data cube by some identifier attributes, we still protect the privacy of data owner.

Without loss of generality, we assume the parity query has the form:

$$PQ_v = Pr[g(v) = 1] = \frac{|\{i \in [n] : g(k,v) = 1\}|}{|D|}$$

So if without noise, the more $Pr[g(v)]$ is near 1, the more hypothesis v is correct.

6.2 Releasing algorithm

For two neighboring databases, the GS is $1/n$. The algorithm is like the algorithm of interval query, we use c to present the perturbed total number of each cell.

Steps:
1. get the original data distribution ordered by cell.
2. perturb the probability mass in every cell to c with Lap(b).
3. if $c > 0$, output c*n and the cell.

By Theorem 5.1, we get the algorithm above preserves α-differential privacy.

Because $GS = 1/n$, according to Theorem 4.2, we know the $\hat{D}$ is (ϵ, δ)-useful if $n \ge \frac{ln\frac{\beta}{\delta}}{\alpha\epsilon}\beta$.

6.3 Learning algorithm of parity

In addition to guaranteeing (ϵ, δ)-usefulness for parity queries, the released data can be also used to learn parity function effectively. [2, 12] proposed a learning algorithm for parity function in the presence of random classification noise and adversarial noise. We briefly introduce the algorithm according to [2] and show the learning result in section 8.

We assume the data is uniformly distributed and the noise added is classification noise. For the cell that satisfies the parity vector r, even if we add Laplace noise to the cell by the mechanism of differential privacy, it is not noise for the parity query. We use η_1 to present the original noise rate. Our algorithm add η_2 noise rate to the data. The total noise rate η in the released data is $\eta_1 + \eta_2$. Therefore, $\eta_2 \le \epsilon$, $\eta \le \eta_1 + \epsilon$.

THEOREM 6.1 ([2]). *the length-d parity problem, for noise rate η equal to any constant less than 1/2, can be solved with number of samples and total computation-time $2^{O(d/logd)}$.*

Therefore, if $\eta < 1/2$, the $\hat{D}$ is learnable. $\eta \le \eta_1 + \epsilon$ and $n \ge \frac{ln\frac{\beta}{\delta}}{\alpha\epsilon}\beta$, so if $n \ge \frac{ln\frac{\beta}{\delta}}{\alpha(1/2-\eta_1)}\beta$, the $\hat{D}$ is learnable.

LEMMA 6.1 ([2]). *Take s samples from a database with parity vector v and noise rate η, then*

$$Pr[\oplus_{i=1}^{s} x_i^{(d)} = \oplus_{i=1}^{s} g(i,v)] = \frac{1}{2} + \frac{1}{2}(1-2\eta)^s.$$

note that each $x_i^{(d)}$ may not be the correct answer of g(i,v) because of noise.

We give the algorithm as follows:

Learning steps for parity vector:
For every target bit j of vector v, $j = 1 : d-1$
1. Draw s samples from the released data where $s > d-1$ to avoid linear dependence.
2. for $k = 1 : d-1$,

   choose a record x_i where $x_i^{(k)} = 1$, remove x_i from the samples and XOR the rest samples with x_i, $x = x \oplus x_i$.

3. if in left samples doesn't exists $x^{(j)} = 1$, then goto 1; else after the elimination of step 2, we discard those samples which become 0, then randomly draw a sample t from the left samples.

$$Pr[v^{(j)} = 1] = Pr[t^{(j)} = 1] = \frac{1}{2} + \frac{1}{2}(1-2\eta)^{d-2}$$

with the lemma described above, we know that every $x^{(j)}$ has probability $\frac{1}{2} + \frac{1}{2}(1-2\eta)^{d-2}$ to be correct. Therefore, with probability $\frac{1}{2} + \frac{1}{2}(1-2\eta)^{d-2}$, we output the correct bit of vector v.

7. MAX QUERY

Query max() is also a distributive query for no matter directly posing the query max() to all data or first partitioning the data then posing max() to the max results of each partition, the results are the same. However, to release data for max query we get two difficulties. First, The returned value of max() is the real data existing in the database. If we release it, at least we will expose one record in the database. Second, For two neighboring database D_1 and D_2 which only differ in one record, if the differing record of D_2 is larger than $max(D_1)$, then releasing D_2 is telling public that D_2 has the max record while D_1 doesn't. Third, by adding Laplace noise to the original answer, the returned result could be meaningless when beyond the domain of data.

To tackle the difficulties mentioned above, we use the releasing algorithm for queries whose output range are on limited domain and specify r^+ as $[r_i - \epsilon, r_i + \epsilon]$ and r^- as $[min(D), r_i - \epsilon]$ where r_i is the data x_i. Without loss of generality, we design the max query as follows.

DEFINITION 7.1. *Max query is to get the max value in the data with high probability,* $Q(D, r) = Pr[Max = r]$ *where* $r \in \mathcal{R}$ *and* $x \in \mathcal{D}$.

Then we use the releasing algorithm of section 4.1.2 for max query which is proven to be α-differentially private. If it's uniformly distributed, the released $\hat{D}$ is (ϵ, δ)-useful while $n \geq \frac{\beta}{\alpha\epsilon} ln \frac{\beta}{\epsilon\delta(e^{\alpha}+1)^2}$ and $\delta < \frac{1}{e^{\alpha}+1}$. Experiment result is given in section 8.

8. EXPERIMENT

8.1 Interval Query

We use the CENSUS data(http://www.ipums.org) to implement the algorithm and evaluate the performance. It has 1 million tuples and 4 attributes: Age, Education, Occupation and Income, whose domain sizes is 79, 14, 23 and 100. All experiments run on a computer with Intel P8600($2 * 2.4$ GHz) CPU and 2GB memory.

8.1.1 1D interval query

We use the Income attribute to simulate 1D interval queries.

$\beta = 100; \epsilon = 0.02; \alpha = 0.05; \delta = 0.05$; With theorem 5.2, we get $n \geq 7.6 * 10^5$. The data set has 1 millon rows, n=1000000, So it satisfies the lower bound.

Elapsed time is 0.155 seconds. Figure 2, 3 show the original and released distribution. X-axis is the Income, y-axis is the count number of each income over the total domain. We released 100 versions of the data by our algorithm and pose 1000 uniformly generated queries for each version of data. The average error is $1.2 * 10^{-5}$. We count the times each error happens in Figure 4. X-axis is the query error ϵ, y-axis is the probability of each error. Figure 5 shows the actual error and the ϵ that we chose at the beginning of our algorithm, which is $\epsilon = 0.02$. The vertical line is bound of ϵ.

8.1.2 2D Interval query

We use the Age and Income attribute to simulate 2D rectangle queries.

$\beta_1 = 80; \beta_2 = 100; \beta = \beta_1 * \beta_2 = 8000; \epsilon = 0.01; \alpha = 0.05; \delta = 0.05$; With theorem 5.2 ,we get $n \geq 1.92 * 10^8$. Because we only get 1 million points in census database, we multiply each cell by 200, so we get $2*10^8$ points, $n = 2*10^8$.

Elapsed time is 0.309 seconds. Figure 6, 7 show the original and release distribution. X-axis is income, y-axis is age, z-axis is the count number of each pair of income and age over the total domain. We released 1 version of data and asked 10^6 uniformly generated queries to compare the query answer with the original answer. The average error is $7.0 * 10^{-6}$. We count the times that each error happens in Figure 8. X-axis is the query error ϵ, y-axis is the probability of each error. Figure 9 shows the actual error and the ϵ we chose at the beginning of our algorithm, which is $\epsilon = 0.01$. The vertical line is bound of ϵ.

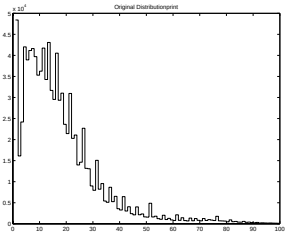

Figure 2: Original Distribution of Income

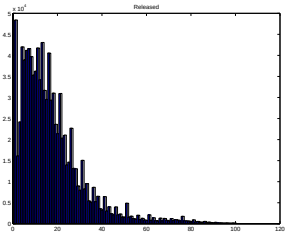

Figure 3: Released Distribution of Income

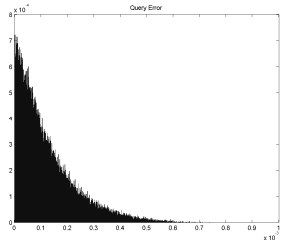

Figure 4: Query Error of 1D Interval

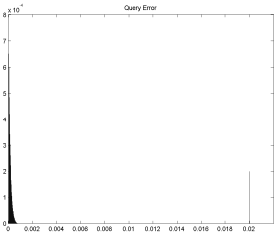

Figure 5: Actual Error and Bound

Table 2: Influence of parameter α

α	Average Error(10^{-6})	Max Error(10^{-6})
1	0.69	34.92
0.5	1.3	88.7
0.1	5.8	453.7
0.05	12	623.9
0.02	31	1724
0.01	63.5	4284
0.005	120	9107

8.1.3 Influence of Parameters

Note that the actual error of 2D interval is less than the actual error of 1D. The reason is that as n becomes larger, $GS = 1/n$, GS become smaller, so the noise added becomes smaller. Therefore, as n becomes larger, the actual error becomes smaller.

We take the example of 1D interval to evaluate the influence of parameter α. We fix n=10^6. Then for each α, we compute the average error and max error. We generate 100 versions of data and randomly ask 1000 uniformly generated queries on each version of data, showed in table 2.

From the table, we can see that when $\alpha = 1$, the noise added is sparse, and the error almost disappears. However, according to the definition of differential privacy, $Pr[\mathcal{A}(D_1) = \hat{D}]/Pr[\mathcal{A}(D_2) = \hat{D}_2] \leq exp(1) = 2.718$, privacy can be violated easily. When α becomes smaller, $exp(\alpha)$ gets near 1 while the average error become bigger. But because the actual error is far less than ϵ, it allows us to take small α. For example, when $\alpha = 0.1$, $exp(\alpha) = 1.1$ which is very close to

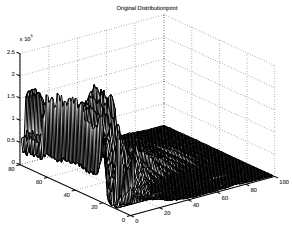

Figure 6: Original Distribution of Income and Age

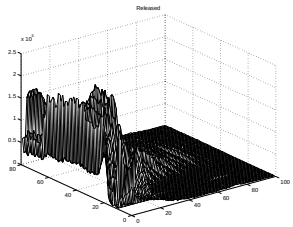

Figure 7: Released Distribution of Income and Age

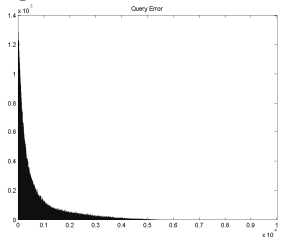

Figure 8: Query Error of 2D Interval

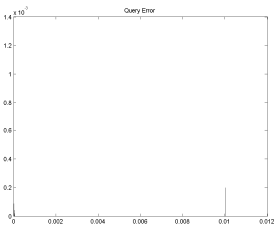

Figure 9: Actual Error and Bound

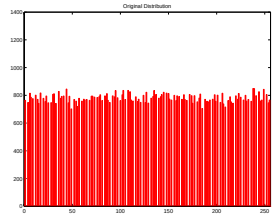

Figure 10: Original Distribution

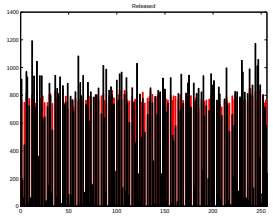

Figure 11: Released Distribution

1, and the average error is $5.8 * 10^{-6}$, meaning that in 10^6 data, we get the average count error 5.8 for randomly generated query. Informally, we suggest that take α less than 0.1.

8.2 Parity function

We uniformly generate 10^5 data with 8 attributes. We design the hidden vector v to be 0000011. Attribute 8th is the result of parity function. Therefore, we have 2^8 kinds of binary strings at most which conform to the parity vector. We don't add noise to the original data, meaning that the original noise rate $\eta_1 = 0$. $\alpha = 0.01$.

Figure 10, 11 show the result of experiment. Elapsed time is 0.17 seconds. Noise rate caused by the algorithm is $\eta_2 = 0.0655$. So the total noise rate $\eta = \eta_1 + \eta_2 = 0.0655$, which is less than 0.5.

We use the learning algorithm to learn the parity class from the released data. First, we draw 32 samples from the data, after the elimination, we draw one sample from the left samples, which may be correct with probability $\frac{1}{2} + \frac{1}{2}(1-2\eta)^{d-2}$. We do this 1000 times for each bit, then get an average probability from the 1000 values for each bit. The learning result

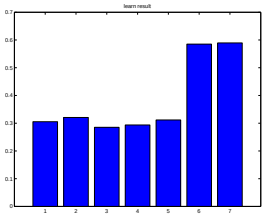

Figure 12: Learning result of parity

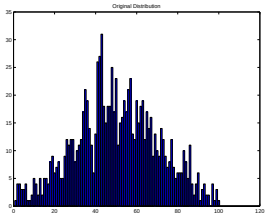

Figure 13: Original distribution for n=1000

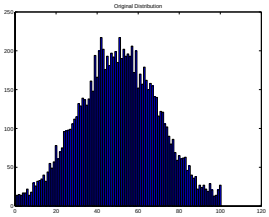

Figure 14: Original distribution for n=10000

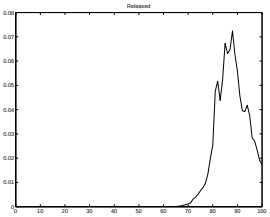

Figure 15: Output probability for n=1000

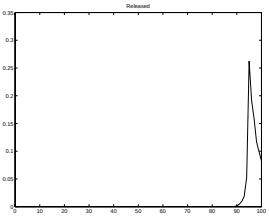

Figure 16: Output probability for n=10000

is shown in Figure 12, from which we can see the original parity vector "0000011" is learned from the released data.

8.3 Max query

We generate data from gaussian distribution in the discretized domain $[1, 2, \cdots, 100]$, with $mean = 50$, $variance = 20$. $\alpha = 0.05$, $\epsilon = 5$. Figure 13, 14 show the original distribution for data size n=1000, 10000. Elapsed time is 0.02 and 0.18 seconds respectively. Figure 15, 16 show the released distribution. We can see that as the data size become larger, the probability to output the correct answer increases. The probability $1 - \delta$ with which the result is in the range $[Max - \epsilon + 1, Max]$ is 0.1523 and 0.9116 respectively.

9. CONCLUSIONS AND FUTURE WORKS

Although recent researches are mostly focused on interactive model for privacy protecting, non-interactive is more convenient for data user and data curator for it only needs to change or publish the data once and it can be used for any users. We proposed the general method to release data for distributive queries. The key to non-interactive release data is to protect the privacy and guarantee the utility of released data. To achieve the contradicting goals, it needs a probability based solution that output the data dynamically with the understanding that the more the output is close to the true answer, the more the probability should be.

With the mentioned criterions of privacy and utility satisfied, it's possible to release data for other form of query. In this paper, we use the (ϵ, δ)-usefulness to measure the utility of data instead of learnability for concept class. Besides parity function, we will investigate whether the released data of this method is PAC learnable for any other concept class in future works.

10. REFERENCES

[1] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the sulq framework. pages 128–138. PODS, 2005.

[2] A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the Acm*, 50(4):506–519, 2003.

[3] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. *Stoc'08: Proceedings of the 2008 Acm International Symposium on Theory of Computing*, pages 609–617, 2008. 14th Annual ACM International Symposium on Theory of Computing MAY 17-20, 2008 Victoria, CANADA.

[4] S. Chawla, C. Dwork, F. McSherry, and K. Talwar. On the utility of privacy-preserving histograms. UAI, 2005.

[5] C. Dwork. Differential privacy. In *in ICALP*, pages 1–12. Springer, 2006.

[6] C. Dwork. Differential privacy. *Automata, Languages and Programming, Pt 2*, 4052:1–12, 2006. Bugliesi, M Prennel, B Sassone, V Wegener, I 33rd International Colloquium on Automata, Languages and Programming JUL 10-14, 2006 Venice, ITALY.

[7] C. Dwork. Differential privacy: A survey of results. In M. Agrawal, D.-Z. Du, Z. Duan, and A. Li, editors, *TAMC*, volume 4978 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2008.

[8] C. Dwork. Differential privacy: A survey of results. *Theory and Applications of Models of Computation, Proceedings*, 4978:1–19, 2008. Agrawal, M Du, DZ Duan, ZH Li, AS 5th International Conference on Theory and Applications of Models of Computation APR 25-29, 2008 Xian, PEOPLES R CHINA.

[9] C. Dwork, K. Kenthapadi, F. McSherry, and I. Mironov. Our data, ourselves: Privacy via distributed noise generation. pages 486–503. Springer, 2006.

[10] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. pages 265–284. Proceedings of the 3rd Theory of Cryptography Conference, 2006.

[11] D. Feldman, A. Fiat, H. Kaplan, and K. Nissim. Private coresets. *Stoc'09: Proceedings of the 2009 Acm Symposium on Theory of Computing*, pages 361–370, 2009. 41st Annual ACM Symposium on Theory of Computing MAY 31-JUN 02, 2009 Bethesda, MD.

[12] V. Feldman, P. Gopalan, S. Khot, and A. K. Ponnuswami. New results for learning noisy parities and halfspaces. *47th Annual IEEE Symposium on Foundations of Computer Science, Proceedings*, pages 563–572, 2006. 47th Annual IEEE Symposium on Foundations of Computer Science OCT 21-24, 2006 Berkeley, CA.

[13] P. Fischer and H. U. Simon. On learning ring-sum-expansions. *Siam Journal on Computing*, 21(1):181–192, 1992.

[14] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey on recent developments. *ACM Computing Surveys*, in press.

[15] D. Helmbold, R. Sloan, and M. K. Warmuth. Learning integer lattices. *Siam Journal on Computing*, 21(2):240–266, 1992.

[16] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? *Proceedings of the 49th Annual Ieee Symposium on Foundations of Computer Science*, pages 531–540, 2008. 49th Annual Symposium on Foundations-of-Computer-Science OCT 25-28, 2008 Philadelphia, PA.

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

[18] K. M and H. J, W. *Data mining: concepts and techniques, Second Edition*. MorganKaufman, 2006.

[19] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*, page 24, 2006.

[20] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. *2008 Ieee 24th International Conference on Data Engineering, Vols 1-3*, pages 277–286, 2008. 24th IEEE International Conference on Data Engineering APR 07-12, 2008 Cancun, MEXICO.

[21] F. McSherry and K. Talwar. Mechanism design via differential privacy. *48th Annual Ieee Symposium on Foundations of Computer Science, Proceedings*, pages 94–103, 2007. 48th Annual IEEE Symposium on Foundations of Computer Science OCT 20-23, 2007 Providence, RI.

[22] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. Knowl. Data Eng.*, 13(6):1010–1027, 2001.

[23] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. volume 10, pages 571–288, 2002.

[24] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, 2002.

[25] X. Xiao and Y. Tao. Output perturbation with query relaxation. pages 857–869. Proceedings of the 34th International Conference on Very Large Data Bases (VLDB), 2008.