PRISM: Prefix-Sum based Range Queries Processing Method under Local Differential Privacy

Yufei Wang [†1], Xiang Cheng [†2*]
[†]State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing, China
Email: [†]{wyfs4321[1], chegnxiang[2]}@bupt.edu.cn

***Abstract*—Range query over data cubes is a powerful tool for online analytical processing (OLAP). In this paper, we focus on answering range queries while satisfying local differential privacy (LDP). The key technical challenges come from the problem of noise aggregation and the curse of high dimensionality: multiple LDP noise will be aggregated when answering range queries and collecting high-dimensional data under LDP will further degrade the utility of the results. To this end, we present a novel method called Prefix-Sum based Range QuerIes ProceSsing Method (PRISM). Its main idea is to selectively collect a few prefix-sums in a data-dependent way, and answer range queries over prefix-sum-based cubes based on which any range query can be processed by using constant pieces of prefix-sums. In PRISM, we first alleviate the problem of noise aggregation by proposing a LDP mechanism called Range based Randomized Response (RRR) and a new type of prefix-sums-based cube called Grained Prefix-Sum (GPS) cube. We then alleviate the curse of high dimensionality by proposing a Data-Dependent Selective Prefix-Sum Collection Strategy (DELFT). We conduct experiments on both real-world datasets and synthetic datasets. Experimental results confirm the effectiveness of PRISM over existing methods.**

I. Introduction

For directing market management and guiding scientific research, enterprises have been collecting users' data and building data warehouses. A typical approach for data analysis is to construct data cubes by extracting the marginals of interested attributes from the data warehouses and perform range queries over these data cubes. The range query applies an aggregation operation over all selected cells of the data cube where the selection is specified by providing ranges of values for ordinal attributes. However, to meet users' expectation of their privacy, enterprises must provide rigorous privacy guarantees on how their data are collected and analyzed.

Differential privacy (DP) [1] has been increasingly accepted as the state-of-the-art standard for protecting individual privacy in the centralized setting where a trusted data aggregator obtains data from all individuals and injects noise in the analytical process to protect privacy. However, in the absence of such a trusted data aggregator, users are reluctant to contribute their private data. Considering this situation, technique for satisfying DP in the local setting, which is called local differential privacy (LDP) [2], has been studied. LDP, which has been adopted in several real-world applications, including Google Chrome browser [3] and macOS [4], enables the gathering of statistics while preserving the privacy of every user, without relying on the trust in a single data aggregator.

Several methods [5], [6], [7] are proposed for answering range queries over multi-dimensional data under LDP. However, these methods cannot cope well with the problem of noise aggregation: when answering range queries over the data cubes which are constructed under LDP, the LDP noise in each cell will be aggregated. Noise aggregation makes multiple LDP noise be concentrated in the results of range queries, leading to uncontrollable error. In addition, if the local data is high-dimensional, e.g., more than 10 attributes, constructing data cubes from local high-dimensional data via existing LDP mechanisms [8], [9], [10] will result in excessive LDP noise which further degrades the utility of the results.

To alleviate the problem of noise aggregation and the curse of high dimensionality, we propose a new method for answering range queries, called **P**refix-Sum based **R**ange Quer**I**es Proce**S**sing **M**ethod (PRISM). Its main idea is to selectively collect the prefix-sums [11] among the coarse-grained local ordinal attributes in a data-dependent way and construct a prefix-sum-based cube for answering range queries. In particular, the prefix-sum is a kind of auxiliary information where any range query can be processed by using constant pieces of prefix-sums.

In PRISM, we first consider how to calculate the prefix-sums under LDP and put forward a specialized LDP mechanism called **R**ange based **R**andomized **R**esponse (RRR) for collecting the essential information that is used when calculating prefix-sums. RRR judges whether user' values belong to the range that corresponds to a prefix-sum and uploads only a binary result. Based on the binary values uploaded by all users, the data aggregator calculates the corresponding prefix-sums. Comparing with the straightforward approach that calculating the prefix-sums from the perturbed local data value, RRR simplifies the process of calculating prefix-sums under LDP and guarantees the utility of collected prefix-sums. Moreover, we design a new type of prefix-sum-based cube called **G**rained **P**refix-**S**um (GPS) cube, which is a multi-dimensional array that stores prefix-sums at a coarser-grained level. On the one hand, since the prefix-sum stores the sum of elements in a certain range, any range query can be processed over the GPS cube by using constant cells. On the other hand, by

The corresponding author of this paper is Prof. Xiang Cheng.

using binning to partition the domains of attributes into coarse granularities, the GPS cube can deal with the large domains of attributes. Furthermore, for constructing the GPS cubes over high-dimensional data without adding excessive LDP noise, we come up with a **D**ata-D**E**pendent Se**L**ective Pre**F**ix-sum Collec**T**ion Strategy (DELFT). DELFT alleviates the curse of dimensionality by avoiding collecting the 2-D prefix-sums that can be inferred easily. Specifically, in DELFT, the data aggregator only collects the 1-D prefix-sums of all attributes and the 2-D prefix-sums among several attributes, which are carefully selected in a data-dependent way, then estimates the GPS cubes based on these prefix-sums.

The main contributions of this work are summarized as follows:

- We present the **P**refix-Sum based **R**ange Quer**I**es Proce**S**sing **M**ethod (PRISM) for answering range queries under LDP which can alleviate the problem of noise aggregation and the curse of high-dimensionality. We show that PRISM satisfies ϵ-local differential privacy.
- We develop a LDP mechanism for calculating the prefix-sums under LDP called **R**ange based **R**andomized **R**esponse (RRR) which simplifies the problem of calculating prefix-sums under LDP and guarantees the utility of collected prefix-sums.
- We design a new type of prefix-sum-based cube called **G**rained **P**refix-**S**um (GPS) cube where any range query can be processed by using constant pieces of cells of GPS cubes. The GPS cubes can also deal with the large domain of attributes.
- We put forward a **D**ata-D**E**pendent Se**L**ective Pre**F**ix-sum Collec**T**ion Strategy (DELFT) to construct GPS cubes over high-dimensional data with high accuracy.
- To evaluate the performance of PRISM, we conduct extensive experiments on both real-world and synthesized datasets. The experimental results demonstrate that PRISM outperforms the baseline methods.

The remainder of this paper is organized as follows. Section 2 reviews the related literature. Section 3 provides the preliminaries. Section 4 describes several baseline methods. Section 5 gives the details of PRISM. Section 6 shows our experimental results. Finally, Section 7 concludes our work.

II. Related Work

In this section, we review the related topics.

A. LDP Mechanisms

Since the conception of LDP is formalized [2], LDP mechanisms for collecting local private data have been extensively studied [12], [8], [3], [9], [10]. Randomized response (RR) [12] can be adopted to collect single-dimensional binary attributes under LDP. For collecting multi-valued attributes, Kairouz et al. [8] extend RR and put forward Generalized randomized response (GRR). Targeting the same problem, Erlingsson et al. [3] propose RAPPOR by applying RR to a Bloom filter. However, both GRR and RAPPOR will suffer from huge accuracy decrease when the domains of multi-valued attributes are large. To tackle this problem, Bassily and Smith [9] propose a random matrix projection-based approach called Succinct Histogram (SH). Wang et al. [10] optimize SH by adopting binary local hashing resulting in a new Optimized Local Hashing (OLH), which provides much better accuracy.

B. Estimating Marginals Privately

A λ-D marginal, i.e., the joint distribution of λ attributes, is the essential for constructing data cube. In the centralized setting, Hardt et al. [13] propose Multiplicative Weights Exponential Mechanism (MWEM), which initializes a full contingency table that is uniform and updates the table via obtaining a noisy answer to the selected marginal. Gupta et al. [14] describe a learning based method that views the marginal as a function on queries. However, these methods become computationally unfeasible in high-dimensional setting. The state-of-the-art method in the centralized setting is proposed by Qardaji et al. [15] called Priview, which computes marginals for a number of strategically chosen sets of attributes that called views, and then uses these views to estimate any desired λ-D marginal. Zhang et al. [5] extend Priview into the local setting and come up with Consistent Adaptive Local Marginal (CALM).

C. Answering range queries

For answering range queries over data cubes in the centralized setting under DP, Hay et al. [16] propose a method which is able to deal with single-dimensional range queries via hierarchical intervals. Qardaji et al. [17] optimize the hierarchical intervals by choosing a proper branching factor. For the multi-dimensional range queries, Xiao et al. [18] develop a data publishing technique called Privelet that ensures DP while providing accurate answers for range queries. For minimizing the max error in range queries, Ding et al. [19] present a general noise control framework which carefully selects a subset of cells in a cuboid to inject laplace noise and computes the rest cells from them. In order to be capable of scaling to large multi-dimensional domains, McKenna et al. [20] propose High-Dimensional Matrix Mechanism (HDMM), which answers queries based on an implicit matrix representation. However, all these methods mentioned above cannot be applied in the local setting. Graham et al. [21] first formalize the problem of answering range queries under LDP and propose a method based on the discrete haar wavelet transform. For the same problem, Wang et al. [6] put forward the Hierarchical-Interval Optimized Mechanism (HIO) for answering single-dimensional range queries. To perform range queries over multi-dimensional data, Yang et al. [7] propose Hybrid-Dimensional Grids (HDG). However, the performance of HDG relies heavily on the choice of granularities of the grids. For the high-dimensional setting, HDG will choose much coarser granularities which lead to a large amount of non-uniformity error.

III. Preliminaries

In this section, we first introduce some preliminaries, then show the problem statement of answering range queries under LDP and the general framework for solving this problem.

A. Range Queries over Data Cubes

1) Data Cube: A data cube is a λ-dimensional ('λ-D') array which is used to represent data (sometimes called facts) along some measures of interest. Every dimension represents a separate attribute and the cells in the cube contain all possible marginals among λ attributes.

For a batch of records that contain d ordinal attributes, a data cube can be constructed among any λ attributes for exploring their correlations. Specifically, the dimensions, denoted by $D_1 \cdots D_\lambda$, are a series of quasi-identifiers that are used to characterize entities from different angles. We show an example of constructing a 2-D data cube from local records in Figure 1.

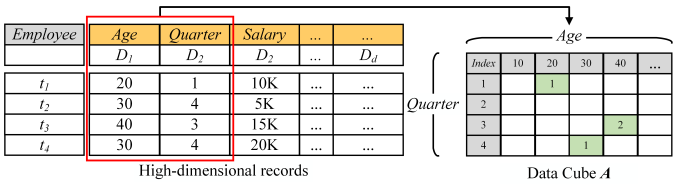


Fig. 1. Constructing data cube from local records

In addition, there are many variants of data cube proposed to cope with different data analysis tasks [11], [10], [7]. Although their structures are different from the canonical data cube, these variants all contain the collection of marginals. For convenience, we will refer to them as data cubes in this paper.

2) Range Query: A range query applies an aggregation operation over all selected cells of a data cube where the selection is specified by providing ranges of values for oridnal dimensions. Let a λ-D array A of size $|M_1| \times |M_2| \times \cdots \times |M_\lambda|$ denote the λ-D data cube, where M_i represents the domain of D_i and $|M_i|$ represents the number of distinct values contained in D_i. The problem of computing a range query in A can be formulated as follows:

$$Sum\left(l_1 : h_1, \cdots, l_\lambda : h_\lambda\right) = \sum_{i_1=l_1}^{h_1} \cdots \sum_{i_\lambda=l_\lambda}^{h_\lambda} A[i_1, \cdots, i_\lambda]. \quad (1)$$

Example 1. *For the data cube A shown in Figure 1, the answer for a range query is $Sum\left(30:40, 3:4\right) = A(30,3) + A(30,4) + A(40,3) + A(40,4) = 0+1+2+0 = 3$, which means that there are 3 employees aged thirties or forties that have received salary in the second half year.*

B. Prefix-Sum Cube

Given λ attributes, the prefix-sum cube is a λ-D array which has the same structure as the corresponding data cube. The cells in prefix-sum cube contain a kind of auxiliary information called prefix-sum [11], which can be computed from the marginals among λ attributes. Based on the prefix-sum cube, any range query can be processed by using constant pieces of prefix-sums, irrespective of the size of the sub-cube circumscribed by a query.

Suppose there is a λ-D data cube A of size $|M_1| \times |M_2| \times \cdots \times |M_\lambda|$. The prefix-sum cube $\mathcal{P}$ is another λ-d array which is used to store various precomputed prefix-sums of A. We will precompute, for all $0 \le x_j < |M_j|$ and $j \in [0, \lambda-1]$,

$$\begin{aligned}\mathcal{P}\left[x_1, x_2, \cdots, x_\lambda\right] &= Sum\left(0:x_1, 0:x_2, \cdots, 0:x_\lambda\right) \\ &= \sum_{i_1=0}^{x_1} \cdots \sum_{i_\lambda=0}^{x_\lambda} A[i_1, \cdots, i_\lambda]. \end{aligned} \quad (2)$$

The theorem below provides how range query can be computed from up to 2^λ appropriate prefix-sums of $\mathcal{P}$. The left hand side of (3) specifies a range query. The right hand side of (3) consists of 2^λ additive terms, each of which is from an element of $\mathcal{P}$ with a sign '+' or '-' defined by the product of all $s(i)$'s. For notational convenience, let $\mathcal{P}\left[x_1, x_2, \cdots, x_\lambda\right] = 0$ if $x_j = -1$ for some $j \in [0, \lambda-1]$.

Theorem 1. *For all $j \in [0, \lambda-1]$, let*

$$s\left(j\right) = \begin{cases} 1, & \text{if } x_j = h_j \\ -1, & \text{if } x_j = l_j - 1 \end{cases},$$

Then, for all $j \in [0, \lambda-1]$,

$$\begin{aligned} &Sum\left(l_1 : h_1, \cdots, l_d : h_d\right) = \\ &\sum_{\forall x_j \in \{l_j-1, h_j\}} \left\{ \left[\prod_{i=1}^{d} s\left(i\right) * \mathcal{P}\left[x_1, x_2, \cdots, x_d\right] \right] \right\}. \end{aligned} \quad (3)$$

Example 2. *Figure 2 shows an example of the prefix-sum cube $\mathcal{P}$ with $\lambda = 2$, $|M_1| = |M_2| = 4$, and the geometrical explanation for the process of answering range queries over $\mathcal{P}$. Suppose there is a range query $Sum\left(1:3, 1:3\right)$, the answer can be obtained by the values of 4 cells in $\mathcal{P}$: $Sum(1:3, 1:3) = Sum(0:3, 0:3) - Sum(0:3, 0:0) - Sum(0:0, 0:3) + Sum(0:0, 0:0) = \mathcal{P}(3,3) - \mathcal{P}(0,3) - \mathcal{P}(3,0) + \mathcal{P}(0,0)$.*

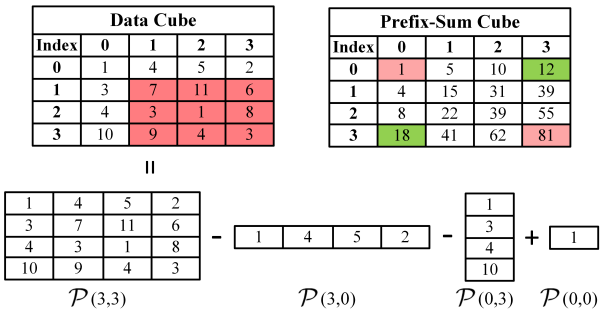


Fig. 2. Answering Range Query over $\mathcal{P}$

C. Local Differential Privacy

1) Definition: Local Differential Privacy (LDP) requires each user to locally perturb his or her record $t_i (1 \le i \le n,$

where n is the number of users) to obtain a randomized version $\tilde{t}_i$, and send $\tilde{t}_i$ to the data aggregator. The perturbation in LDP ensures that the data aggregator cannot distinguish the exact record t_i from the perturbed one $\tilde{t}_i$ with high confidence. Formally, LDP is defined as follows.

Definition 1 (Local Differential Privacy)**.** *Given a privacy budget ϵ, a randomized algorithm $\mathcal{R}$ satisfies ϵ-LDP, if for any two tuples t and $t' \in dom(\mathcal{R})$, and for any possible output $\tilde{t} \in Range(\mathcal{R})$, we have*

$$Pr[\mathcal{R}(t) = \tilde{t}] \leq e^{\epsilon} \cdot Pr[\mathcal{R}(t') = \tilde{t}],$$

where the probability space is over the coin flips of $\mathcal{R}$.

Intuitively, a smaller ϵ leads to stronger privacy guarantee and lower data accuracy.

2) LDP Implementation Mechanism: We present the LDP mechanism that is used in our paper.

Randomized Response. Randomized Response (RR) [12] is used for estimating the statistics of a binary attribute under LDP. Assume that each user has the value of a binary attribute (say, 0 or 1). The data aggregator aims to compute the number of users possessing '0' without learning the true value of any user. To achieve this, each user sends his or her value v with probability p and the other value with q to the data aggregator. More formally, the perturbation function is defined as

$$Pr\left[\tilde{v} = y\right] = \begin{cases} p = \frac{e^{\epsilon}}{e^{\epsilon}+1} & \text{if } y = v \\ q = \frac{1}{e^{\epsilon}+1} & \text{if } y \neq v \end{cases}.$$

Then the data aggregator counts how many times '0' is collected (denoted by c) and computes the unbiased noisy count $c' = \frac{c-nq}{p-q}$ where n is the total number of users.

The result is an unbiased estimation of the true frequency and the variance for this estimation is

$$\mathbf{Var}_{RR} = \frac{e^{\epsilon}}{n \cdot (e^{\epsilon} - 1)^2} \tag{4}$$

Besides RR, there are several LDP mechanisms proposed for collecting multi-valued data.

Generalized Randomized Response. Generalized randomized response(GRR) [8] extends RR for supporting multi-valued attribute. In GRR, each user with value $v \in D$ sends the true value v with probability p, and with probability q sends a randomly chosen $v' \in D$ s.t. $v' \neq v$. GRR has the perturbation function which is similar to RR:

$$Pr\left[\tilde{v} = y\right] = \begin{cases} p = \frac{e^{\epsilon}}{e^{\epsilon}+|D|-1} & \text{if } y = v \\ q = \frac{1}{e^{\epsilon}+|D|-1} & \text{if } y \neq v \end{cases}.$$

The estimated count is unbiased and has variance

$$\mathbf{Var}_{GRR}(\epsilon) = \frac{(m - 2 + e^{\epsilon})}{n \cdot (e^{\epsilon} - 1)^2} \tag{5}$$

Optimal Local Hashing. The Optimal Local Hashing(OLH) [10] is a state-of-the-art LDP mechanism which deals with a large domain. Each user first projects his or her record to a smaller domain g via a random matrix selected from a universal hash function family and then applies GRR[8] to the projected record. In OLH, the variance as a function of g is minimized when $g = e^{\epsilon} + 1$:

$$\mathbf{Var}_{OLH} = \frac{4e^{\epsilon}}{n \cdot (e^{\epsilon} - 1)^2} + f \tag{6}$$

where f is the frequency of the value being estimated.

As shown in [10] that when collecting the multi-valued attribute with a small domain (where $m < 3e^{\epsilon} + 2$), GRR is better; and for a large domain, OLH is preferable.

Theorem 2 (Parallel Composition)**.** *Given q random algorithms $\mathcal{R}_i (1 \leq i \leq q)$ that access different tuples $t_i (1 \leq i \leq q)$, each of which satisfies ϵ_i-LDP, then the combination of their outputs satisfies ϵ-LDP, where $\epsilon = max\,(\epsilon_1, \cdots, \epsilon_q)$.*

Inspired by parallel composition, the user dividing strategy [7], which collects multiple pieces of information without allocating privacy budget, has been put forward. This strategy has been widely used to avoid the increase of estimation variance which is caused by overallocation of privacy budget. That is, when multiple pieces of information are needed, the data aggregator divides users into groups and then gathers information from each group with privacy budget ϵ. According to parallel composition, the entire process still satisfies ϵ-LDP. In this paper, we will also adopt the user dividing strategy in PRISM.

D. Answering Range Queries under LDP

1) Problem Statement: Suppose there are n users, and each user owns a record with d ordinal attributes. The untrusted data aggregator collects these d-dimensional data under LDP and constructs data cubes for answering any range query.

2) General Framework: Based on the analysis of existing methods [6], [7], we give the general framework for answering range queries under LDP, which consists of a **cube construction** phase and a **query answering** phase. Specifically, the data aggregator constructs the data cubes in the first phase, then answers the range queries in the second phase. We show the process of the general framework in Figure 3.

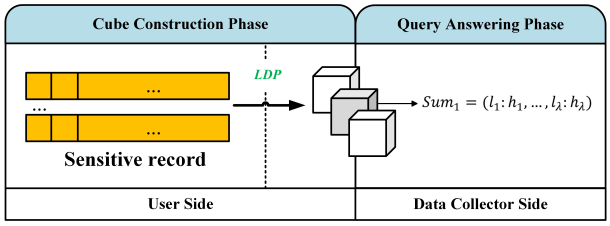


Fig. 3. Answering Range Queries under LDP

By answering range queries under LDP via the general framework, the variance of result is $\mathbf{Err}(\tilde{Sum}) = \omega \cdot \mathbf{Err}_{ldp}$, where $\mathbf{Err}_{ldp}$ is the variance of the LDP noise introduced in each cell of data cube and ω is a multiplicative factor that represents the number of noisy cells that are aggregated.

$\mathbf{Err}_{ldp}$ is amplified by ω times when the perturbed values in the selected cells are aggregated during the query answering

phase. Let m denote the number of specified values for each queried attribute, ω is polynomial [5], [7] or polylogarithmic [6] in the cardinality of m. It means that when m is large, the ω will grow rapidly and lead to a large amount of noise in the answer.

For a certain LDP mechanism, both the large domain and the high dimension of data will affect the scale of $\mathbf{Err}_{ldp}$. On the one hand, the large domain increases the difficulty of estimating the distribution of attributes. On the other hand, the high dimension increases the number of marginals collected, which means a fixed number of users will be divided into too many groups. According to (4), a small number of users in each groups will lead to a large variance, resulting in inaccurate statistical results.

IV. Baseline Methods

In the following, we will describe three baseline methods that can be used for answering range queries under LDP, including CALM, HIO, HDG. In particular, CALM is the state-of-the-art protocol for releasing marginal under LDP which can be extended to answer range queries; HIO attempts to reduce the number of noisy cells aggregated by proposing a hierarchy-based data cube; HDG is the state-of-the-art method for answering range queries based on a hybrid data cube that consists of fine-grained 1-D marginals and coarse-grained 2-D marginals of the local data.

A. CALM

Consistent Adaptive Local Marginal (CALM) [5] is a protocol for releasing marginals under LDP which can be extended to answer range queries. CALM first divides users into C_d^2 groups and then gathers the 2-D marginals between each pair of attributes from corresponding groups. Based on these 2-D marginals, CALM estimates data cubes via maximum entropy optimization. All range queries can be answered over corresponding data cubes.

Instead of collecting the high-dimensional marginals for constructing a data cube directly, CALM reduces the number of groups divided from C_d^λ to C_d^2 and results in slightly lower LDP noise added. However, CALM fails to solve the problem of noise aggregation. Moreover, in the high-dimensional setting where $d > 10$, the number of marginals to be collected still increases rapidly as d grows. Both problems mentioned above lead to a large amount of noise.

B. HIO

The Hierarchical-Interval Optimized Method (HIO) [6] proposes a hierarchy-based variant of data cube for answering range queries, which is inspired by the tree structure with a branching factor b. Given an attribute with the domain M, the data aggregator constructs a 1-D hierarchy for the attribute. To be specific, the 1-D hierarchy can be viewed as a perfect b-ary tree with $\lfloor \log_b |M| \rfloor$ one-dim levels. Each node of the b-ary tree corresponds to an interval, and has b children corresponding to b equally sized sub-intervals. In the 1-D hierarchy, each one-dim level represents a different granularity for the entire domain and needs to be collected separately. Then, for λ attributes, the data aggregator constructs a λ-D hierarchy. A level in the λ-D hierarchy is actually a $(\lambda-1)$-D hierarchy. Thus there are $\lfloor \log_b^\lambda |M| \rfloor$ one-dim levels in the λ-D hierarchy. To construct this λ-D hierarchy, the users are divided into $\lfloor \log_b^\lambda |M| \rfloor$ groups, where each group reports the marginal for a corresponding one-dim level. Based on the λ-D hierarchy, any range query that involves λ attributes can be decomposed into sub-queries on these intervals and computed from up to $\omega = \log_b^\lambda |M|$ cells, resulting less noisy being aggregated.

Comparing with CALM, HIO reduces the multiplicative factor ω from $O(m^\lambda)$ to $O(\log_b^\lambda m)$, which alleviates the problem of noise aggregation to some extent. However, in the setting where the data aggregator needs to construct any λ-D hierarchy from d attributes, users are divided into $\lfloor \log_b^d |M| \rfloor$ groups. This results in too few users in each group, which will incur a high magnitude of LDP noise added.

C. HDG

Hybrid-Dimensional Grids (HDG) [7] is the state-of-the-art method for answering range queries under LDP. Its main idea is to carefully use binning to partition attributes' domains into two grids with different granularities g_1 and g_2, which capture the d finer-grained 1-D grids on the distribution of each individual attribute and the C_d^2 coarse-grained 2-D grids on the marginal of all attribute pairs. In HDG, the users are randomly divided into $d + C_d^2$ groups, each of which corresponds to one of these grids. Finally, the data aggregator constructs C_d^2 2-D data cubes and d 1-D data cubes based on the marginals collected from corresponding groups. HDG can answer all 2-D range queries over these data cubes and then estimate the answer of higher dimensional range queries from the answers of the associated 2-D range queries.

HDG attempts to tackle the problem of noise aggregation by restricting the upper limit of the number of cells in data cubes through partitioning attributes' domains into coarser-grained grids. However, according to the guideline proposed in [7], high dimensionality will result in much coarser-grained grids, causing unacceptable non-uniformity error.

D. Summary of Baseline Methods

By summarizing the baseline methods introduced above, we find that these methods have two limitations: On the one hand, although HIO and HDG attempt to tackle the problem of noise aggregation, these baseline methods still cannot eliminate the impact of the increase of query range on the utility of the results. On the other hand, when using the baseline methods to answering range queries over high-dimensional data, there are a large number of marginals to be collected for constructing the data cubes, which makes the number of users in each group too small to accurately estimate the marginals.

V. Solution

In this section, we present our method for answering range queries under LDP which alleviates the problem of noise aggregation and the curse of the high dimensionaity.

A. Overview

As shown in Section IV, the baseline methods fail to overcome the problem of the noise aggregation and the curse of high dimensionality. To this end, we present a method called **P**refix-Sum based **R**ange Quer**I**es Proce**S**sing **M**ethod (PRISM). Different from the baseline methods which answer range queries over marginal-based data cubes, PRISM selectively collects some prefix-sums rather than the marginals in a data-dependent way and answers range queries over the prefix-sum-based cubes.

PRISM contains three key building blocks: a LDP mechanism called **R**ange based **R**andomized **R**esponse (RRR) which calculates prefix-sums from local data; a data model called **G**rained **P**refix-**S**um (GPS) cube which stores prefix-sums at a coarser-grained level for alleviating the problem of noise aggregation and dealing with the large domains of local data; a strategy for constructing GPS cubes in the high-dimensional setting called **D**ata-D**E**pendent Se**L**ective Pre**F**ix-sum Collec**T**ion Strategy (DELFT), which takes all 1-D prefix-sums and the 2-D prefix-sums among several selected attributes as a base set and estimates any GPS cubes based on the base set. In particular, the attributes are carefully selected from all attributes based on their 1-D prefix-sums in a data-dependent way. Besides, to make PRISM consistently effective, we provide guidelines for properly choosing the granularity of GPS cubes and the size of the base set, respectively.

Following are the details of PRISM:

Phase 1. Suppose there are d private attributes in each record, the data aggregator first randomly divides users into d groups, each of which corresponds to an attribute. Then, the data aggregator calculates the coarse-grained 1-D prefix-sums from corresponding groups via RRR with privacy budget $\epsilon/2$. Finally, the data aggregator constructs d temporary GPS cubes, which contain the 1-D prefix-sums of each attribute.

Phase 2. Based on these temporary 1-D GPS cubes, the data aggregator selects k attributes according to the guideline in DELFT.

Phase 3. The data aggregator re-divides the users into C_k^2 groups and collects the coarse-grained 2-D prefix-sums among the selected attributes from these groups via RRR with privacy budget $\epsilon/2$. Then, the data aggregator takes all 1-D prefix-sums and the 2-D prefix-sums collected as a base set.

Phase 4. For any range query that involves λ attributes, the data aggregator estimates the corresponding λ-D GPS cube based on the relevant prefix-sums in the base set, and answers the range query over the GPS cube.

In the following, we first introduce the LDP mechanism RRR in Section V-B. Then, in Section V-C we elaborate the GPS cube. Finally, we introduce DELFT in Section V-D.

B. Range based Randomized Response

Answering range queries over the prefix-sum cubes under LDP will only amplify $\mathbf{Err}_{ldp}$ by $\omega = 2^\lambda$ times, which is much less than any baseline method. However, how to calculate prefix-sums under LDP with high utility is still a unsolved problem.

Algorithm 1: RRR in User Side

Input: The number of users n, users' records $t_j = [t_{j1}, t_{j2}, \cdots, t_{j\lambda}]$, where $1 \le j \le n$, the domains of the interested attributes $M_1, M_2, \cdots, M_\lambda$, privacy budget ϵ.

Output: Pertrubed binary values $\tilde{tr_j^I} = RRR(t_j, \epsilon)$.

Let $\mathcal{M} = \{M_1 \times M_2 \times \cdots \times M_\lambda\}$ **for** $I \in \mathcal{M}$ **do**
  for $0 \le j < n$ **do**
    Initialize $inRange = True, tr_j^I = 0$;
    for $1 \le k < \lambda$ **do**
      if $t_{jk} \notin [0, i_k]$ **then**
        $inRange = False$
    if $inRange$ **then**
      $tr_j^I = 1$
    else
      $tr_j^I = 0$
    Generate a perturbed value $\tilde{tr_j^I}$ such that

$$Pr\left[\tilde{tr_j^I} = 1\right] = \begin{cases} p = \frac{e^\epsilon}{e^\epsilon + 1} \text{ if } tr_j^I = 1 \\ q = \frac{1}{e^\epsilon + 1} \text{ if } tr_j^I = 0 \end{cases}$$

    Send $\tilde{tr_j^I}$ and I to data aggregator.

A straightforward approach for calculating prefix-sums under LDP is to first collect users' local data via existing LDP mechanisms [8], [10] and build a data cube. Then, the data aggregator computes prefix-sums from the data cube by summing certain cells, as mentioned in Section III-B. However, the cells in data cube contain LDP noise. The noise will be aggregated when calculating the prefix-sums, leading to a large error in the answers of range queries.

To guarantee the utility of prefix-sums, we put forward a LDP mechanism called **R**ange based **R**andomized **R**esponse (RRR). The main idea of this mechanism is to calculate the prefix-sums directly by collecting the necessary binary range information from the local data rather than deriving the prefix-sums from the noisy data cube.

We first describe how RRR works in both user side and data aggregator side. For ease of illustration, we take a 1-D prefix-sum cube $\mathcal{P}$ with the size of $|M|$ as an example. Each cell $\mathcal{P}[i]$ contains a prefix-sum that circumscribes a range $[0, i]$, where $0 \le i < |M|$. For each range, the j-th user ($0 \le j < n$) first checks whether his or her private value belongs to the range $[0, i]$ and outputs a binary range indicator tr_j^i. If the value belongs to the range, user sets $tr_j^i = 1$, else $tr_j^i = 0$. Then, the j-th user perturbs all $|M|$ binary range indicators tr_j^i via RR respectively and reports the perturbed $\tilde{tr_j^i}$ to the data aggregator. After receiving the perturbed $\tilde{tr_j^i}$ from all users, the data aggregator calculates the noisy prefix-sums $\mathcal{P}[i] = \sum_{j=0}^{n} \tilde{tr_j^i}$. However, due to using RR to ensure

Algorithm 2: RRR in Data Aggregator Side

Input: Prefix-sum cube indexes $I \in \mathcal{M}$, perturbed binary value $\tilde{tr}_j^I$ where $0 \leq j < n$

Output: Estimated prefix-sum cube $\mathcal{P}$.

Initialize prefix-sum cube $\mathcal{P}$;

Aggregate the perturbed records: $\mathcal{P}[I] = \sum_{j=1}^{n} \tilde{tr}_j^I$;

Post-process the value of cells in $\mathcal{P}$;

Return $\mathcal{P}$.

privacy, the noisy prefix-sum $\mathcal{P}[i_2]$ may smaller than $\mathcal{P}[i_1]$ $(0 \leq i_i < i_2 < |M|)$, which leads to inconsistency among cells and violates the prior knowledge that the values in prefix-sum cube are in ascending order. To improve the utility, the data aggregator post-processes the noisy prefix-sum cube to remove the inconsistency. Specifically, for any cell $\mathcal{P}[i_i]$ in the noisy prefix-sum cube $\mathcal{P}$, the data aggregator must make sure

$$0 \leq \mathcal{P}[i-1] \leq \mathcal{P}[i] \leq \mathcal{P}[i+1] \leq n. \tag{7}$$

Algorithms 1 and 2 provide the details of RRR in user side and aggregator side respectively. In the user side, to upload the perturbed binary range indicator among λ interested attributes, RRR takes the domain of these λ attributes, the user' records about these attributes and the privacy budget as inputs. In the data aggregator side, to estimated the prefix-sums among λ interested attributes, RRR takes the indexes of prefix-sum cube and the perturbed binary range indicators $\tilde{tr}$ as inputs. In particular, we use a vector $I = [i_1, i_2, \cdots, i_\lambda]$ to represent the index of a certain cell in a prefix-sum cube.

Finally, we theoretically prove that RRR satisfies LDP.

Proof. Given any two different users' records t, t', any perturbed binary range indicator $\tilde{tr} \in \{0, 1\}$, and the privacy budget $\epsilon' = \frac{\epsilon}{|\mathcal{M}|}$ which is allocated for each collection, we need to prove that

$$\left| \prod_{I \in \mathcal{M}} \frac{Pr[RRR(t, \epsilon') = \tilde{tr}]}{Pr[RRR(t', \epsilon') = \tilde{tr}]} \right| \leq e^{\epsilon}$$

Since the local users only upload a series of binary range indicator to the data aggregator, we have

$$\begin{aligned} &\left| \prod_{I \in \mathcal{M}} \frac{Pr[RRR(t, \epsilon') = \tilde{tr}]}{Pr[RRR(t', \epsilon') = \tilde{tr}]} \right| \\ &= \left| \prod_{I \in \mathcal{M}} \frac{Pr[RR(inRange, \epsilon') = \tilde{tr}]}{Pr[RR(inRange', \epsilon') = \tilde{tr}]} \right| \\ &\leq \frac{p^{|\mathcal{M}|}}{q^{|\mathcal{M}|}} = e^{|\mathcal{M}|\epsilon'} = e^{\epsilon}. \end{aligned} \tag{8}$$

Therefore, we have $\left| \prod_{I \in \mathcal{M}} \frac{Pr[RRR(t, \epsilon') = \tilde{tr}]}{Pr[RRR(t', \epsilon') = \tilde{tr}]} \right| \leq e^{\epsilon}$. As such, RRR satisfies ϵ-LDP.

C. Grained Prefix-Sum (GPS) Cube

In RRR, the privacy budget ϵ is partitioned evenly for collecting the prefix-sums contained in each cell of a prefix-sum cube. When attributes' domains are relatively large, it will bring excessive LDP noise. To deal with the large domains of attributes, we propose an alternative of prefix-sum cube called **G**rained **P**refix-**S**um (GPS) Cube. The GPS cube is a multi-dimensional array that stores the prefix-sums of coarse-grained attributes, where each dimension of the array corresponds to an attribute whose domain is partitioned into coarser granularity g via binning.

1) Choosing the Granularity: Since the domains of attributes in GPS cube are partitioned into coarser grids, values with in the same cell are collected together, which may lead to the occasion that a cell in GPS cubes is partially included in a query. Therefore, the data aggregator must answer the range queries over GPS cubes based on the assume that values within the same cell have a uniform distribution, resulting in the non-uniformity error.

The granularity g of the attributes can affect the utility of the GPS cube: a small g will result in a large amount of non-uniformity error due to the uniformity assumption, while a large g cannot reduce the LDP noise added when calculating prefix-sums. To deal with this problem, we put forward a guideline for choosing the optimal g. The guideline aims to find a g to minimize the sum of the LDP noise and non-uniformity error.

On the one hand, there are 2^λ cells included when answering a range query over the GPS cube, since the noise in each cell is $\frac{e^{\epsilon/g^\lambda}}{n \cdot \left(e^{\epsilon/g^\lambda} - 1\right)^2}$, then the aggregated LDP noise is $\frac{2^\lambda \cdot e^{\epsilon/g^\lambda}}{n \cdot \left(e^{\epsilon/g^\lambda} - 1\right)^2}$. On the other hand, we assume that the non-uniformity error is $\left(\frac{\alpha}{g}\right)^2$ [7], where $\alpha = 0.2$. Finally, by taking the partial derivative of $\frac{2^\lambda \cdot e^{\epsilon/g^\lambda}}{n \cdot \left(e^{\epsilon/g^\lambda} - 1\right)^2} + \left(\frac{\alpha}{g}\right)^2$ with respect to g, we find the optimal $g = \sqrt{n \cdot \frac{\alpha\epsilon}{2^{\lambda-1}}}$.

2) Constructing GPS Cube: For constructing a GPS cube about λ interested attributes with domain M. The data aggregator first derives the granularity g from the guideline and assigns it to local users. The users compress the domain of their local attributes from $|M|$ to g and send the binary range indicators of these coarser-grained attributes via RRR. After receiving these range information, the data aggregator calculates the prefix-sums and constructs a GPS cube with size of g^λ. Figure 4 shows the process of constructing the GPS cube with $\lambda = 2, |M| = 4, g = 2$.

D. Data-Dependent Selective Prefix-Sum Collection Strategy

To answer λ-D range queries, baseline methods [5], [7] first collect the marginals among all attribute pairs for answering 2-D range queries, and estimate the answers of λ-D range queries from the answers of associated 2-D range queries. However, in the high-dimensional setting, the number of marginals to be collected increases rapidly as d grows. It makes the users be divided into too many groups and incurs a large amount of LDP noise. To mitigate the impact of high dimensionality, we answer λ-D range queries via a **D**ata-D**E**pendent Se**L**ective Pre**F**ix-sum Collec**T**ion Strategy (DELFT).

In fact, the 2-D prefix-sums among the attribute pairs whose correlations are weak can be inferred easily without

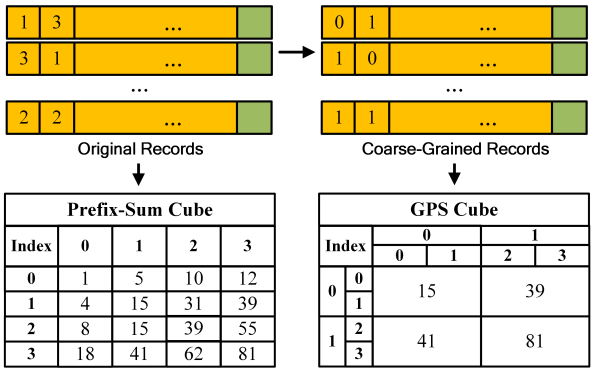


Fig. 4. Constructing the GPS Cube

collection. For example, if two attributes are independent, the normalized 2-D prefix-sums among them can be calculated by multiplying their normalized 1-D prefix-sums. Our DELFT reduces unnecessary collections by calculating only the 1-D prefix-sums of all attributes and the 2-D prefix-sums among a few selected attributes whose related 2-D prefix-sums are hard to be inferred.

In DELFT, the data aggregator goes through two rounds of prefix-sum calculations. In the first round, the data aggregator calculates the 1-D prefix-sums of all attributes. In the second round, the data aggregator calculates the 2-D prefix-sums among several attributes, which are carefully selected based on the 1-D prefix-sums. The data aggregator then takes all 1-D prefix-sums and 2-D prefix-sums as a base set and estimates any λ-D GPS cube based on the base set for answering λ-D range queries.

There are three key issues to discuss about DELFT: how to evaluating the attributes, how to choose the size of base set, and how to estimated the GPS cubes based on the base set. We will discuss these issues, respectively.

1) Evaluating the Attributes: We judge whether the associated 2-D prefix-sums of an attribute are easily inferred by calculating the entropy of the attribute. Given an attribute D that has $|M|$ distinct values and its 1-D distribution P, which can be converted from prefix-sums, the entropy H of D can be calculated by:

$$H(D) = -\sum_{i=1}^{|M|} P(D=i)\log_2 P(D=i). \tag{9}$$

The entropy of an attribute' distribution is a concept in informatics which indicates its uncertainty and information capacity. If the entropy of an attribute is small, it means that the 2-D prefix-sums between this attribute and others are relatively certain and easier to be inferred without further collection. On the contrary, if the entropy of an attribute is relatively high, the 2-D prefix-sums between this attribute and others are more uncertain and worth being collected in the subsequent collection. We explain it through a concrete example:

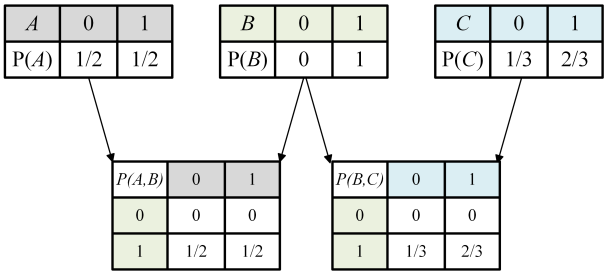


Fig. 5. Evaluating the attributes

Example 3. *Suppose there are three binary attributes A, B, C and their 1-D distributions shown in Figure 5, we attempt to identify which 2-D prefix-sums between these attributes are worth calculated based on the entropies of their 1-D distributions:*

$$H(A) = -\frac{1}{2}\cdot\log_2\frac{1}{2} - \frac{1}{2}\cdot\log_2\frac{1}{2} = 1,$$
$$H(B) = -0\cdot\log_2 0 - 1\cdot\log_2 1 = 0,$$
$$H(C) = -\frac{1}{3}\cdot\log_2\frac{1}{3} - \frac{2}{3}\cdot\log_2\frac{2}{3} = 0.918.$$

Since $H(A) > H(C) > H(B)$, we select A and C and calculate the 2-D prefix-sums between A and C via RRR instead of the prefix-sums related to B. This decision can easily be proven right, since the 2-D distributions related to B can be derived from the 1-D distribution easily, which has been shown in Figure 5, and based on the 2-D distribution we can infer the 2-D prefix-sums related to B directly. However, the 2-D prefix-sums related to A and C are uncertain and required to be calculated in the second round of collection.

2) Choosing the Size of Base Set: After calculating the entropy of all attributes, we sort these attributes in descending order according to their entropies and select the top-k attributes. k is an important parameter which represents the size of the base set: a small k results in the inability to capture the correlations between attributes and causes excessive estimation error; while a large k means there are more prefix-sums to be collected, with greater LDP noise added. We propose a guideline for choosing an appropriate k to minimize the sum of estimation error and LDP noise.

To formalize this optimization problem, we first quantify the estimation error in an informatic way: since the estimation error is depending on the uncertainty of the data distribution, we represent it with the uncertainty of all uncollected 2-D prefix-sums. It can be calculated by subtracting the amount of information contained in the base set from the information capacity of all 2-D prefix-sums. We bound the estimation error as:

$$C_d^2\cdot\log_2 g^2 - \sum_{0\le i<j<k} \tilde{H}\left(d_i, d_j\right), \tag{10}$$

where $C_d^2\cdot\log_2 g^2$ is the information capacity of all 2-D prefix-sums, and $\sum_{0\le x<j<k}\tilde{H}\left(d_i,d_j\right)$ is the amount of

information expected to be obtained by collecting the 2-D prefix-sums between any selected attribute pairs. Note that $\sum_{0\le x<j<k} \tilde{H}(d_i, d_j)$ can be calculated via the maximum entropy model based on 1-D prefix-sums. Then, we quantify the LDP noise of C_k^2 pieces of 2-D prefix-sums by plugging the privacy budget $\epsilon' = \frac{\epsilon}{g^2}$ into (4):

$$\frac{C_k^2 \cdot e^{\epsilon/g^2}}{n \cdot \left(e^{\epsilon/g^2} - 1\right)^2}. \tag{11}$$

Based on (10) and (11), the sum of the estimation error and the amount of LDP noise can be formalized as:

$$C_d^2 \cdot \log_2 g^2 - \sum_{1\le x<j\le k} \tilde{H}(d_i, d_j) + \frac{C_k^2 \cdot e^{\epsilon/g^2}}{n \cdot \left(e^{\epsilon/g^2} - 1\right)^2}. \tag{12}$$

After the first round of collection, we plug all possible k into (12) and find the optimal k which minimizes the entire error.

3) Estimating the λ-D GPS cube: Given a λ-D range query, we estimate the corresponding λ-D GPS cube based on the base set. In PRISM, we use the maximum entropy model to estimate the λ-D prefix-sums contained in the λ-D GPS cube. We can express the estimation problem as the following optimization:

$$\begin{aligned} \max_{p} \quad & -\sum p \log_2 p, \\ \text{s.t.} \quad & A \cdot p = a, \\ & B \cdot p = b, \\ & \sum p = 1, \end{aligned}$$

where p is a $\lambda^d \times 1$ vector representing the unknown λ-D distribution, a is a $d\lambda \times 1$ vector representing the 1-D prefix-sums, b is a $k\lambda^2 \times 1$ vector representing the 2-D prefix-sums, A is a $d\lambda \times \lambda^d$ matrix representing the mapping relationship between a and p, and B is a $k\lambda^2 \times \lambda^d$ matrix representing the mapping relationship between b and p.

Based on the λ-D distribution p, we can derive the λ-D prefix-sums and construct the corresponding GPS cube.

E. Privacy and Utility Analysis

In this section, we first show the privacy guarantee of PRISM, then we analyze the utility of PRISM and compare it with baseline methods from four aspects: LDP noise, noise aggregation, non-uniformity error and estimation error.

1) Privacy Guarantee: It is obvious that PRISM satisfies ϵ-LDP, because there are two rounds of collection in PRISM and in each round the information from each user to the data aggregator goes through RRR with $\frac{\epsilon}{2}$ as privacy budget, without any other information being leaked.

2) LDP Noise: We first focus on the LDP noise $\mathbf{Err}_{ldp}$ added in each cell. Specifically, both the number of prefix-sums to be collected and LDP mechanism used for each collection will affect $\mathbf{Err}_{ldp}$. On the one hand, according to the user dividing strategy, users are divided into groups and each group corresponds to a piece of prefix-sum to be collected. Increasing the number of prefix-sums to be collected will make each group size smaller. Small group size leads to inaccurate statistical results. On the other hand, the LDP mechanism directly affects the LDP noise added in a single collection. In PRISM, by choosing k attributes and collecting the 2-D prefix-sums between these selected attributes, the data aggregator first reduces the number of prefix-sums that is required to be collected to $d + C_k^2$. The data aggregator then collects these prefix-sums via RRR, which simplifies the problem of collecting ordinal data to that of collecting binary data. In contrast, CALM directly collects the 2-D marginals between all possible ordinal attribute pairs via OLH, where the users are divided into C_d^2 groups and the sample size in each group is $\frac{n}{C_d^2}$. HDG is even worse than CALM since it increases the number of marginals that is required to be collected from C_d^2 to $d + C_d^2$. In the high-dimensional setting where $d > 10$, the baseline methods will divide users into too many groups, leading to a large amount of LDP noise.

3) Noise Aggregation: Then we consider the problem of noise aggregation, where $\mathbf{Err}_{ldp}$ is amplified by ω times when selected cells are summed up during answering the range queries under LDP. Specifically, in all baseline methods, the multiplicative factor ω is polynomial [5], [7] or polylogarithmic [6] in the cardinality of m. In CALM, range queries are answered over canonical data cubes, where the multiplicative factor is $\omega = O(m^\lambda)$. HDG attempts to tackle the problem of noise aggregation by restricting the upper limit of the number of cells in data cubes, leading to a multiplicative factor $\omega = O((\frac{m \cdot g_2}{|M|})^\lambda + \lambda \cdot \frac{m \cdot g_1}{|M|})$. Comparing with CALM and HDG, HIO answers range queries over the hierarchy-based data cubes and leads to $\omega = O(\log_b^\lambda m)$ which is polylogarithmic in the cardinality of m. Obviously, in the baseline methods, when the size of the sub-cube circumscribed by the range queries grows, a large amount of noise will aggregate, which degrades the utility of answers. To alleviate the problem of noise aggregation, PRISM answers range queries over the GPS cubes, where all range queries for a given GPS cube can be processed by using constant cells. It means the multiplicative factor $\omega = 2^\lambda$ is a constant regardless of m, which is smaller than any baseline method and leads to a better utility.

4) Non-uniformity Error: The non-uniformity error is caused by the uniform assumption which is involved when answering range queries over the coarse-grained cubes. The coarser the granularity, the larger the non-uniformity error. Here we mainly focus on the non-uniformity error added in HDG and PRISM by comparing the sizes of their granularities in a concrete setting. Given the setting where there are 10^6 records and each of which contains 10 attributes, according to the guideline in Section V-C1, the granularity of GPS cube is $g = 10$. By contrast, in the same setting, according to the guideline in [7], the coarser-grained 2-D data cubes in HDG have a granularity $g_2 = 2$ and the finer-grained 1-D data cubes in HDG have a granularity $g_1 = 4$. Obviously, the granularities in HDG are coarser than that of PRISM, which means the non-uniformity error in PRISM is much lower than HDG.

5) Estimation Error: In CALM, HDG and PRISM, the estimation error is added when the data aggregator answers the λ-D range queries via maximum entropy model or other

estimation methods. The more information the data aggregator takes as input in advance, the more accurate the estimation result is. In CALM and HDG, the data aggregator takes the 2-D marginals between all possible attribute pairs as input to estimate the answers of λ-D range queries; while in PRISM, the data aggregator only takes the prefix-sums in the initial set as input to estimate the λ-D GPS cubes. As a result, the estimation error caused in PRISM is greater than that in the baseline methods. However, DDSP minimizes the sum of estimation error and LDP noise by choosing the base set reasonably, which reduces the influence of estimation errors on the results of range queries.

VI. Experiments

In this section, we first evaluate the performance of PRISM for answering range queries by varying different parameters. We then evaluate the advantages of PRISM' key building blocks. Finally, we evaluate the guidelines for choosing the optimal granularity used in the GPS cube and the guideline for choosing the size k of the base set, respectively.

A. Experimental Settings

Datasets. We make use of two real datasets and two synthetic datasets in our experiments:

- *IPUMS [22]:* A US census dataset from the IPUMS repository, which contains around 3 million records.
- *Adult [23]:* A dataset from the UCI machine learning repository. After removing missing values, the dataset contains around 50 thousands records.
- *Laplace:* A dataset which is synthesized from multivariate laplace distribution with mean 0, standard deviation 1, contains 50 ordinal attributes. The covariance between every two attributes is 0.5.
- *Random:* A dataset which is synthesized from uniform distribution, contains 50 ordinal attributes. The covariance between every two attributes is 0.5.

To experiment with different numbers of users, we generate multiple test datasets from the four datasets with the number of users ranging from 10k to 10M. For evaluating different numbers of attributes and domain sizes, we generate multiple versions of these four datasets with the number of attributes ranging from 10 to 50 and their domain sizes ranging from 10 to 50.

Competitors. We compare PRISM against all the baseline methods including CALM, HIO, HDG. In addition, to evaluate the advantages of the three key building blocks in PRISM, we prepare four variants of PRISM, called NON-RRR, NON-GPS, MIN and MAX. Specifically, for NON-RRR, we collect all prefix-sums via OLH rather than RRR to evaluating the advantage of RRR; for NON-GPS, we collect the prefix-sums via RRR and then calculate the marginals from these prefix-sums and estimate all possible data cubes with DELFT. For MIN, we default to that $k = 0$ and only collect all attributes' 1-D prefix-sums for estimating the GPS cubes. For MAX, we collect the 2-D prefix-sums among all attributes for estimating the GPS cubes.

In particular, in order to reduce the computational cost of the baseline methods, we improve CALM and HIO by compressing the attributes' domain before collecting the local data.

Utility Metric. We use Mean Absolute Error to measure the accuracy of range queries' results: $MAE = AVG\left(\left|\frac{\tilde{Sum}-Sum}{n}\right|\right)$, where n is the total number of records, $\tilde{Sum}$ and Sum are the estimated and true results of a range query, respectively.

Methodology. For each dataset and each method, we choose 100 random range queries with specific query volumes denoted by $vol(q) = \frac{m}{|M|}$, which means the ratio of the specified range to the domain size for each queried attribute. We repeat all experiments 10 times and measure the MAE of their results.

In all subsequent experiments, unless explicitly stated, we use the following default values for other relevant parameters: $\epsilon = 2.0, vol(q) = 0.5, d = 30, |M| = 30, n = 10^5$.

Environment. We implement all the methods in Python. The source code of PRISM is publicly available at [24]. All experiments are conducted on an Intel Core i5 2.50GHz PC with 8GB RAM.

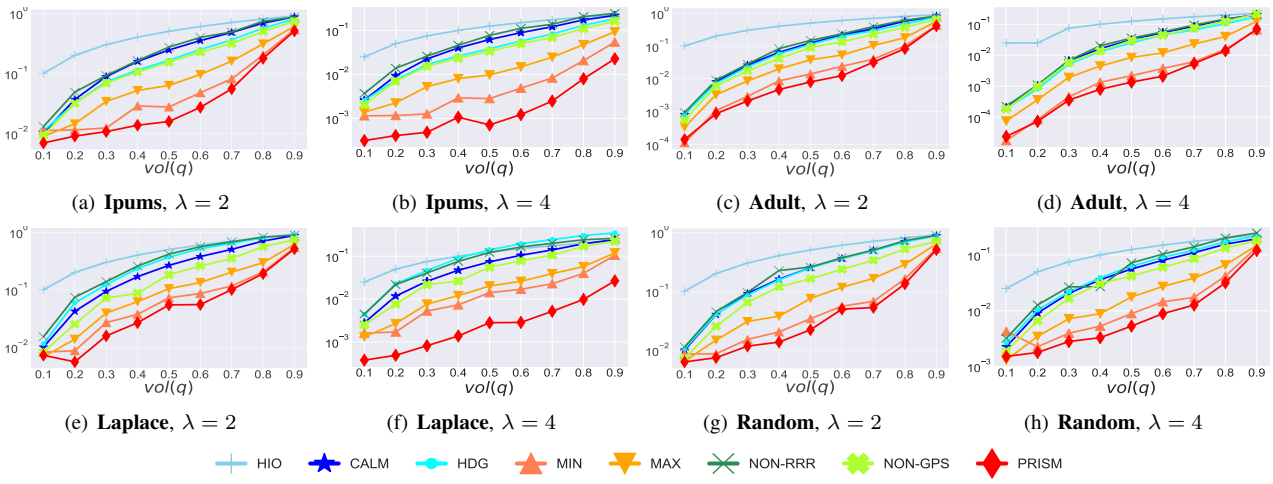
B. Impact of Different Parameters

In this part, we compare different methods under various parameter settings. In general, these parameters including the query volume $vol(q)$ of range queries, the privacy budget ϵ, the number of attributes d, the domain of each attribute M, the total number of users n, and the query dimension λ.

Varying query volume $vol(q)$. Figures 6(a)-6(d) show the impact of $vol(q)$ on the performance of each method. We observe that as $vol(q)$ increases, the MAEs of all methods increase, and PRISM gets the best performance, followed by its variants. Notice that the trend of PRISM's performance as $vol(q)$ grows is consistent with the fact that the variance of range queries' result is constant in PRISM. Specifically, such a constant variance indicates that the ratio of the MAE to the results of range queries is fixed. As $vol(q)$ increases, there are more records involved in the results, which drives the increase of MAE.

Varying privacy budget ϵ. Figures 6(e)-6(h) show the results varying ϵ from 1.0 to 5.0 . It is obviously that PRISM and its variants have a clear advantage over other methods, and PRISM performs better than its variants. We also find that comparing with other methods, the MAEs of PRISM and MIN are more sensitive to the growth of ϵ. The reason can be explained as follows. Since PRISM and MIN alleviate the problem of noise aggregation and the curse of high dimensionality, ϵ becomes a major factor affecting the utility of PRISM and MIN. By contrast, the performance of other methods is affected by multiple factors, and only increasing the privacy budget cannot significantly improve the utility.

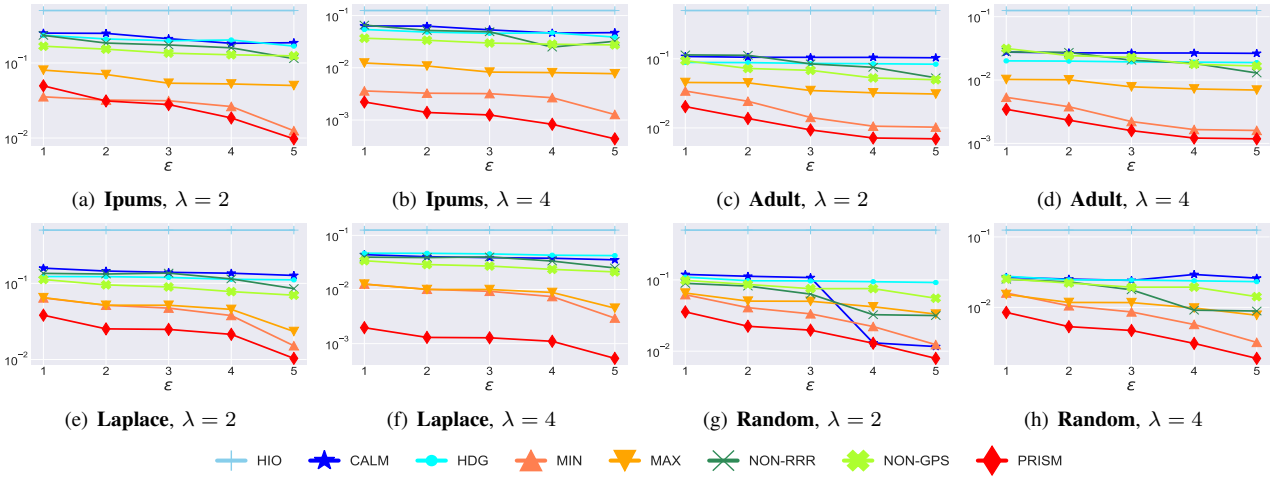
Varying the number of attributes d. Figures **??-??** present the results varying d from 10 to 50 on all datasets. We can observe that PRISM consistently outperforms all other methods. In general, the MAEs of all methods become higher



(a) **Ipums**, $\lambda = 2$ (b) **Ipums**, $\lambda = 4$ (c) **Adult**, $\lambda = 2$ (d) **Adult**, $\lambda = 4$

(e) **Laplace**, $\lambda = 2$ (f) **Laplace**, $\lambda = 4$ (g) **Random**, $\lambda = 2$ (h) **Random**, $\lambda = 4$

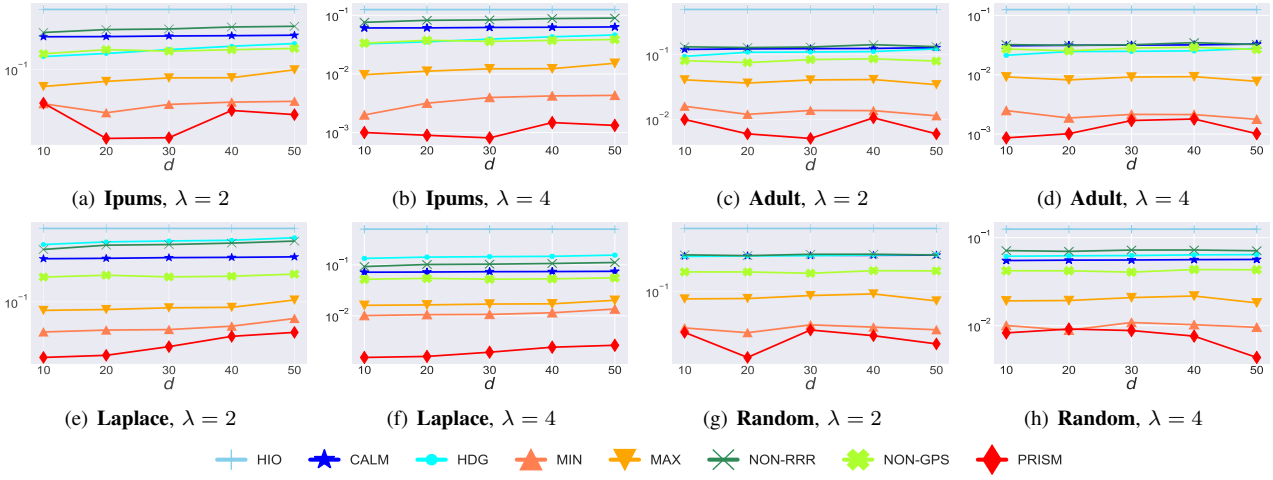
Fig. 6. Varying $vol(q)$ under setting of $\epsilon = 2.0, d = 30, |M| = 30, n = 10^5$.



(a) **Ipums**, $\lambda = 2$ (b) **Ipums**, $\lambda = 4$ (c) **Adult**, $\lambda = 2$ (d) **Adult**, $\lambda = 4$

(e) **Laplace**, $\lambda = 2$ (f) **Laplace**, $\lambda = 4$ (g) **Random**, $\lambda = 2$ (h) **Random**, $\lambda = 4$

Fig. 7. Varying ϵ under setting of $vol(q) = 0.5, d = 30, |M| = 30, n = 10^5$.



(a) **Ipums**, $\lambda = 2$ (b) **Ipums**, $\lambda = 4$ (c) **Adult**, $\lambda = 2$ (d) **Adult**, $\lambda = 4$

(e) **Laplace**, $\lambda = 2$ (f) **Laplace**, $\lambda = 4$ (g) **Random**, $\lambda = 2$ (h) **Random**, $\lambda = 4$

Fig. 8. Varying d under setting of $vol(q) = 0.5, \epsilon = 2.0, |M| = 30, n = 10^5$.

as d increases, which is consistent with our analysis that more marginals or prefix-sums to be collected leads to more LDP noise added in the answer. In addition, we also find there are some outliers where the PRISM's MAEs at $d = 50$ are smaller than those at $d = 40$. This is due to the changes of the size of base set. In particular, as d increases from 40 to 50, the optimal k changes, leading to a smaller size of base set.

Varying the domain of attributes $|M|$. Figures **??-??** study the impact of $|M|$ on the utility of each method. It is obvious that PRISM performs the best among all methods. Moreover, we observe that all methods perform steadily when $|M|$ becomes larger. The reason is that all these methods compress the domain of the attributes by binning, which reduces the noise added when collecting the local data via LDP mechanisms.

Varying the number of users n. For different datasets, we sample data size n as a variable and show the results in Figures **??-??**. As it can be seen, the MAEs of all methods decrease with the increase of the number of users, and PRISM always performs the best. Besides, we observe that there are some outliers at $n = 10^5$ in Figures **??** and **??** where the MIN's MAEs at $n = 10^5$ are larger than those at $n = 5 \times 10^4$. The reason can be explained as follows. MIN only estimates GPS cubes based on 1-D prefix-sums. When the number of users increases, MIN cannot capture the correlations among attributes and leads to unstable results.

Varying query dimension λ. Figures **??-??** show the results varying λ from 1 to 5. We observe that when λ becomes larger, the MAEs of all methods will decrease. The reason can be explained as follows. Due to the error is uniformly distributed in all cells, when the query range $vol(q)$ of each dimension is fixed, the larger λ is, the smaller the proportion of the cells involved in the query to the total cells will be, resulting in the decrease of the total amount of the noise.

C. Performance of Key Building Blocks

In this part, we analyze the advantages of the three key building blocks in PRISM by comparing PRISM with its variants, the results are shown in Figure 6.

By comparing NON-RRR with PRISM, we evaluate the advantage of RRR. It is shown that if the prefix-sums are collected without RRR, there is a sharp decline in the utility of GPS cubes, which makes the performance of NON-RRR even worse than CALM. It indicates that GPS cubes cannot work well independently and RRR is an important building block for calculating prefix-sums under LDP.

The advantage of GPS cubes can be evaluated by comparing NON-GPS with PRISM, since the only difference between these two methods is whether constructing GPS cubes after calculating prefix-sums via RRR. As shown in Figure 6, the performance of NON-GPS is worse than PRISM, which indicates the effectiveness of GPS cubes. Moreover, Figure 6(a)-6(d) show that as $vol(q)$ increases, the MAEs of NON-GPS increase more rapidly than PRISM. It can be explained as follows. NON-GPS cannot deal with the problem of noise aggregation when more cells are involved in the range queries. While PRISM alleviates the problem of noise aggregation by answering range queries over GPS cubes.

We evaluate the advantage of DELFT by comparing PRISM with MIN and MAX. Figure 6 shows that PRISM always performs better than MIN and MAX, which means that estimating GPS cubes over all 1-D prefix-sums and a few carefully selected 2-D prefix-sums will significantly improve the accuracy.

D. Effectiveness of Guidelines in PRISM

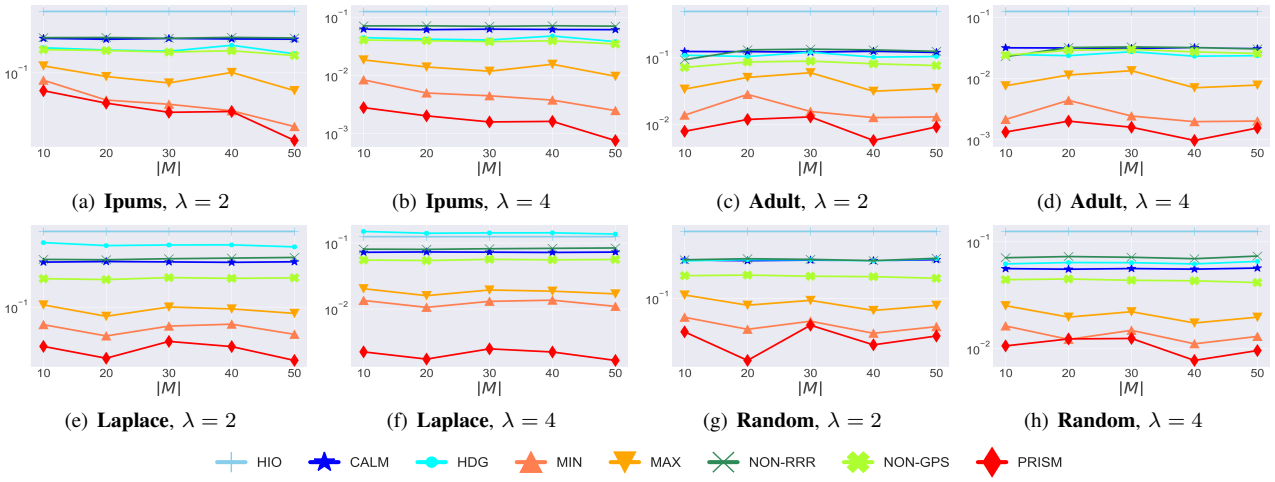
In this part, we evaluate the guidelines for choosing the optimal granularity g used in the GPS cube and the optimal size k of base set in DELFT. We judge whether our guidelines can provide good choices by comparing PRISM with several different implemented versions of PRISM where the granularity g or base set size k is set in advance, and present the results in Figure 12 and Figure 13.

To evaluate the effectiveness of guideline for choosing the granularity g, we first enumerate several possible $g = 5, 10, 15, 20, 25, 30$ for a given domain $|M| = 30$. Then, for each g, we use it as the chosen granularity to implement a version of PRISM. Finally, we compare PRISM with all the implemented versions under the setting where $\epsilon = 2.0, |M| = 30, d = 30, n = 10^5$ and observe their performance as varying $vol(q)$. As shown in Figure 12, we find that PRISM performs reasonably well for most $vol(q)$. Moreover, we notice that for these implemented versions, the best performance is most easily achieved when $10 \leq g \leq 20$, which means the optimal g is located in this interval and our guideline for choosing g can capture it in most situations.

Then, we evaluate the effectiveness of guideline for choosing the base set size k. Similar to the setting above, we enumerate several possible $k = 0, \frac{d}{4}, \frac{d}{2}, \frac{3d}{4}, d$ and implement different versions of PRISM. We compare PRISM with these implemented versions under the setting where $\epsilon = 2.0, vol(q) = 0.5, |M| = 30, n = 10^5$. The results are presented in Figure 13. It is shown that the MAEs of these versions are concave with the increase of k. PRISM outperforms all other versions, which means that the guideline for choosing the size of base set can provide good choices under different settings.

VII. Conclusion

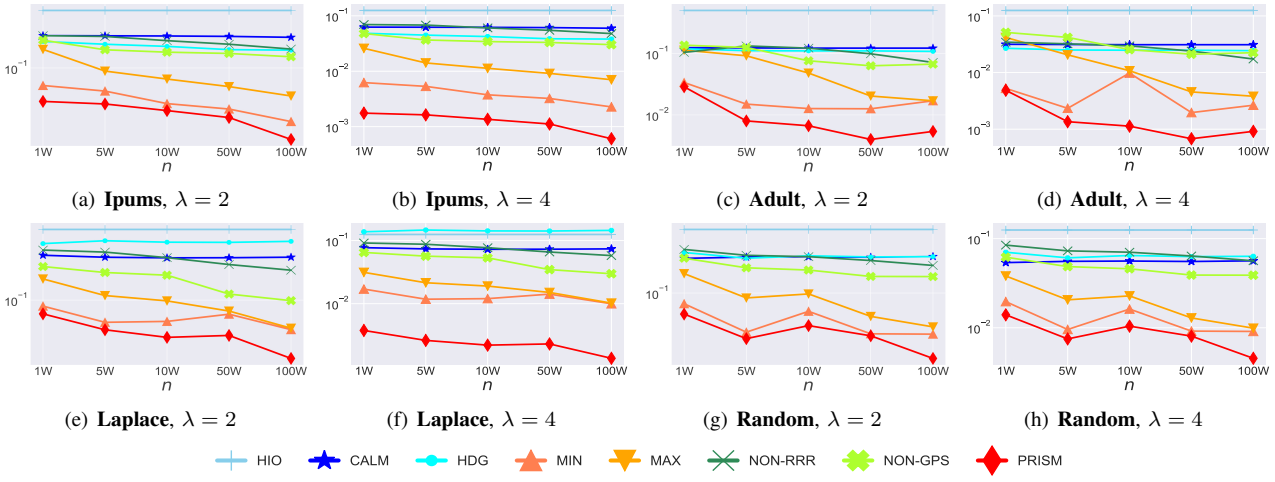
In this paper, we propose PRISM for answering range queries under LDP which alleviates the problem of noise aggregation and the curse of high dimenionality. To calculate prefix-sums under LDP without adding too much noise, we put forward RRR. In addition, for answering range queries without involving too much LDP noise, we propose the GPS cube. Furthermore, to ensure that our method performs well over the high-dimensional data, we come up with DELFT for estimating any high-dimensional GPS cube. Sufficient experiments are conducted to illustrate the effectiveness of PRISM.



(a) **Ipums**, $\lambda = 2$ (b) **Ipums**, $\lambda = 4$ (c) **Adult**, $\lambda = 2$ (d) **Adult**, $\lambda = 4$

(e) **Laplace**, $\lambda = 2$ (f) **Laplace**, $\lambda = 4$ (g) **Random**, $\lambda = 2$ (h) **Random**, $\lambda = 4$

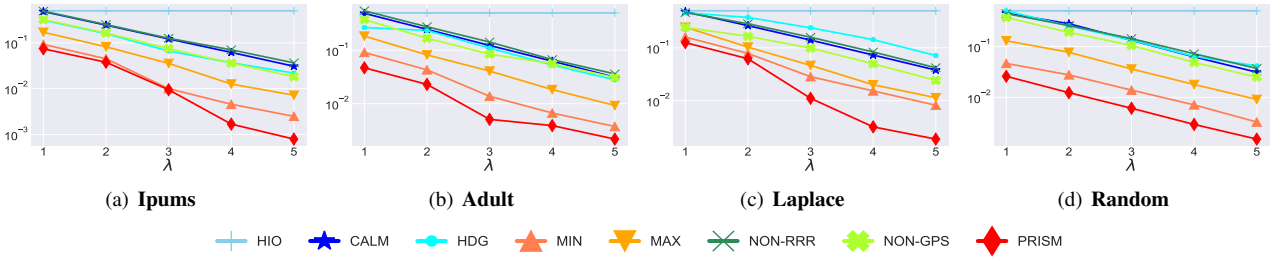
Fig. 9. Varying $|M|$ under setting of $vol(q) = 0.5, \epsilon = 2.0, d = 30, n = 10^5$.



(a) **Ipums**, $\lambda = 2$ (b) **Ipums**, $\lambda = 4$ (c) **Adult**, $\lambda = 2$ (d) **Adult**, $\lambda = 4$

(e) **Laplace**, $\lambda = 2$ (f) **Laplace**, $\lambda = 4$ (g) **Random**, $\lambda = 2$ (h) **Random**, $\lambda = 4$

Fig. 10. Varying n under setting of $vol(q) = 0.5, \epsilon = 2.0, |M| = 30, d = 30$.



(a) **Ipums** (b) **Adult** (c) **Laplace** (d) **Random**

Fig. 11. Varying λ under setting of $vol(q) = 0.5, \epsilon = 2.0, |M| = 30, d = 30, n = 10^5$.

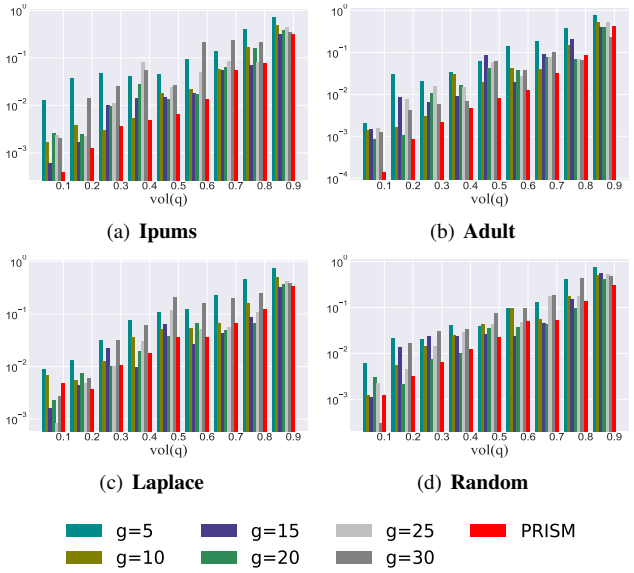


Fig. 12. Comparing PRISM with the variants of different g

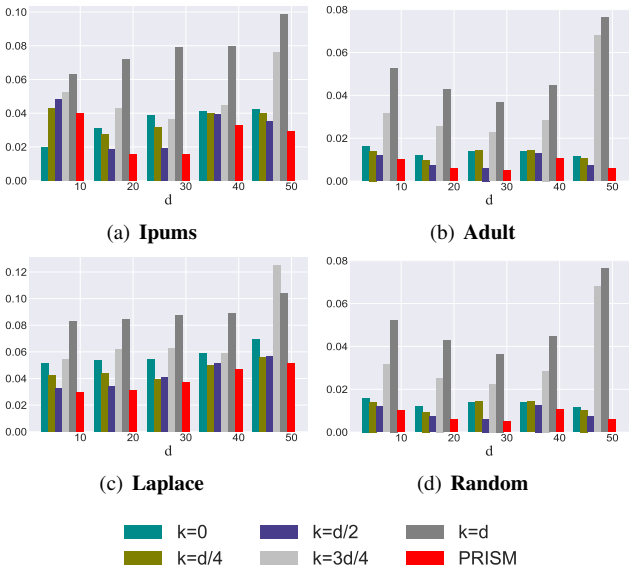


Fig. 13. Comparing PRISM with the variants of different k

REFERENCES

[1] C. Dwork, F. Mcsherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography Conference*, 2006.

[2] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2014.

[3] Ú. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, G. Ahn, M. Yung, and N. Li, Eds. ACM, 2014, pp. 1054–1067. [Online]. Available: https://doi.org/10.1145/2660267.2660348

[4] A. D. P. Team., "Local privacy and statistical minimax rates," 2017.

[5] Z. Zhang, T. Wang, N. Li, S. He, and J. Chen, "CALM: consistent adaptive local marginal for marginal release under local differential privacy," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, D. Lie, M. Mannan, M. Backes, and X. Wang, Eds. ACM, 2018, pp. 212–229. [Online]. Available: https://doi.org/10.1145/3243734.3243742

[6] T. Wang, B. Ding, J. Zhou, C. Hong, Z. Huang, N. Li, and S. Jha, "Answering multi-dimensional analytical queries under local differential privacy," in *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, P. A. Boncz, S. Manegold, A. Ailamaki, A. Deshpande, and T. Kraska, Eds. ACM, 2019, pp. 159–176. [Online]. Available: https://doi.org/10.1145/3299869.3319891

[7] J. Yang, T. Wang, N. Li, X. Cheng, and S. Su, "Answering multi-dimensional range queries under local differential privacy," *Proc. VLDB Endow.*, vol. 14, no. 3, p. 378–390, Nov. 2020.

[8] P. Kairouz, S. Oh, and P. Viswanath, "Extremal mechanisms for local differential privacy," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., 2014, pp. 2879–2887. [Online]. Available: http://papers.nips.cc/paper/5392-extremal-mechanisms-for-local-differential-privacy

[9] R. Bassily and A. D. Smith, "Local, private, efficient protocols for succinct histograms," in *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, R. A. Servedio and R. Rubinfeld, Eds. ACM, 2015, pp. 127–135. [Online]. Available: https://doi.org/10.1145/2746539.2746632

[10] T. Wang, N. Li, and S. Jha, "Locally differentially private frequent itemset mining," in *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. IEEE Computer Society, 2018, pp. 127–143. [Online]. Available: https://doi.org/10.1109/SP.2018.00035

[11] C. Ho, R. Agrawal, N. Megiddo, and R. Srikant, "Range queries in OLAP data cubes," in *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, J. Peckham, Ed. ACM Press, 1997, pp. 73–88. [Online]. Available: https://doi.org/10.1145/253260.253274

[12] S. L. Warner, "Randomized response: a survey technique for eliminating evasive answer bias." *Publications of the American Statistical Association*, vol. 60, 1965.

[13] M. Hardt, K. Ligett, and F. McSherry, "A simple and practical algorithm for differentially private data release," in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., 2012, pp. 2348–2356. [Online]. Available: http://papers.nips.cc/paper/4548-a-simple-and-practical-algorithm-for-differentially-private-data-release

[14] A. Gupta, M. Hardt, A. Roth, and J. R. Ullman, "Privately releasing conjunctions and the statistical query barrier," in *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, L. Fortnow and S. P. Vadhan, Eds. ACM, 2011, pp. 803–812. [Online]. Available: https://doi.org/10.1145/1993636.1993742

[15] W. H. Qardaji, W. Yang, and N. Li, "Priview: practical differentially private release of marginal contingency tables," in *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, C. E. Dyreson, F. Li, and M. T. Özsu, Eds. ACM, 2014, pp. 1435–1446. [Online]. Available: https://doi.org/10.1145/2588555.2588575

[16] M. Hay, V. Rastogi, G. Miklau, and D. Suciu, "Boosting the accuracy of differentially private histograms through consistency," *Proc. VLDB Endow.*, vol. 3, no. 1, pp. 1021–1032, 2010. [Online]. Available: http://www.vldb.org/pvldb/vldb2010/pvldb_vol3/R91.pdf

[17] W. H. Qardaji, W. Yang, and N. Li, "Understanding hierarchical methods for differentially private histograms," *Proc. VLDB Endow.*, vol. 6, no. 14, pp. 1954–1965, 2013. [Online]. Available: http://www.vldb.org/pvldb/vol6/p1954-qardaji.pdf

[18] X. Xiao, G. Wang, and J. Gehrke, "Differential privacy via wavelet transforms," in *Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA*, F. Li, M. M. Moro, S. Ghandeharizadeh, J. R. Haritsa, G. Weikum, M. J. Carey, F. Casati, E. Y. Chang,

I. Manolescu, S. Mehrotra, U. Dayal, and V. J. Tsotras, Eds. IEEE Computer Society, 2010, pp. 225–236. [Online]. Available: https://doi.org/10.1109/ICDE.2010.5447831

[19] B. Ding, M. Winslett, J. Han, and Z. Li, "Differentially private data cubes: optimizing noise sources and consistency," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*, T. K. Sellis, R. J. Miller, A. Kementsietsidis, and Y. Velegrakis, Eds. ACM, 2011, pp. 217–228. [Online]. Available: https://doi.org/10.1145/1989323.1989347

[20] R. McKenna, G. Miklau, M. Hay, and A. Machanavajjhala, "Optimizing error of high-dimensional statistical queries under differential privacy," *Proc. VLDB Endow.*, vol. 11, no. 10, pp. 1206–1219, 2018. [Online]. Available: http://www.vldb.org/pvldb/vol11/p1206-mckenna.pdf

[21] G. Cormode, T. Kulkarni, and D. Srivastava, "Answering range queries under local differential privacy," *Proceedings of the VLDB Endowment*, vol. 12, no. 10, pp. 1126–1138, 2019.

[22] Sobek, Matthew, Ruggles, and Steven, "The ipums project." *Historical Methods*, vol. 32, no. 3, pp. 102–102, 1999.

[23] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[24] "Prism," https://github.com/wyfs4321/PRISM.