Practical Differential Privacy via Grouping and Smoothing

Georgios Kellaris Stavros Papadopoulos

Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
{gkellaris, stavrosp}@cse.ust.hk

ABSTRACT

We address one-time publishing of non-overlapping counts with ϵ-differential privacy. These statistics are useful in a wide and important range of applications, including transactional, traffic and medical data analysis. Prior work on the topic publishes such statistics with prohibitively low utility in several practical scenarios. Towards this end, we present GS, a method that pre-processes the counts by elaborately *grouping* and *smoothing* them via averaging. This step acts as a form of preliminary perturbation that diminishes sensitivity, and enables GS to achieve ϵ-differential privacy through low Laplace noise injection. The grouping strategy is dictated by a sampling mechanism, which minimizes the smoothing perturbation. We demonstrate the superiority of GS over its competitors, and confirm its practicality, via extensive experiments on real datasets.

1. INTRODUCTION

Numerous organizations release proprietary statistics on individuals to third parties. For example, geo-social networks (e.g., Foursquare [1]) may sell user "check-in" summaries to advertising companies. Moreover, hospitals may provide pharmaceutical corporations or universities with patient prescription aggregates for research purposes. However, sensitive information may be inferred from the published data, e.g., that a user visited a night club, or that a patient suffers from a certain disease.

There is a plethora of methods on privacy-preserving data publishing (e.g., [20, 22, 24, 14, 4]). Their target is to hide sensitive information, while retaining the *utility* of the released data. One of the popular paradigms that offers strong privacy is *ϵ-differential privacy* [3]. In particular, it ensures that the adversary extracts roughly the same information, whether the data of a user are incorporated in the aggregate result or not. This implies that the user privacy is protected and, thus, individuals are not discouraged from participating in the statistical analysis.

Proceedings of the VLDB Endowment, Vol. 6, No. 5

The basic ϵ-differential privacy framework entails two entities; a *curator* and a *querier*. The curator is a trusted party which holds a database (e.g., user "check-ins" or patient prescriptions). The querier requests statistical information (e.g., counts) from the curator. The curator executes a mechanism, which perturbs the answers prior to their publication following certain criteria. This *sanitization* process hinders sensitive information disclosure about an individual with strong theoretical guarantees.

1.1 Problem Description and Motivation

We focus on *one-time* publishing of *non-overlapping counts* with ϵ-differential privacy. We clarify the targeted problem with an example. Suppose that the curator collects a database where column j refers to a "check-in" place, record i corresponds to a user, and the value of cell (i, j) is non-zero if the i^{th} user "checked-in" at the j^{th} place at least once. Consider the query "*return all the column counts from the database*", i.e., return the number of users that "checked-in" at each place. Its result indicates the most popular places, and may be useful for location-based advertising.

The above counts are *non-overlapping*, i.e., each cell value affects exactly one count. Moreover, we assume that the curator returns the result count vector only *once*. Equivalently, the curator may publish the result on a web site *once*, and multiple queriers may extract any desired information from the released data (e.g., a subset of counts). There are numerous applications where such data publication is useful, including transactional databases (e.g., for market basket analysis), trajectory databases (e.g., for traffic analysis), medical databases (e.g., for pharmaceutical research), and movie rating databases (e.g., for viewer demographics).

The challenge is to publish the counts with ϵ-differential privacy. Among the plethora of work on ϵ-differential privacy, only two schemes are applicable to our setting: the *Laplace Perturbation Algorithm* [6] (henceforth referred to as LPA), and an adaptation of the *Fourier Perturbation Algorithm* [18] (hereafter referred to as FPA).

LPA achieves ϵ-differential privacy by injecting independent Laplace noise in every count before releasing it. The noise scale is *proportional* to the *sensitivity* of the query, i.e., to the maximum amount of influence of a single user on the result. As such, LPA features satisfactory utility in settings where the sensitivity is low, e.g., in histogram publication where each user affects a *single* count. However, our targeted applications may incur *arbitrary* sensitivity. For example, a user may "check-in" at *any* number of places and, hence, affect any number of counts. In such scenarios, the published data of LPA may become meaningless [21].

FPA mainly targets at *time series* data. Adapting it to our context, every database column corresponds to a timestamp, the result count vector is perceived as a time series, and each user affects an arbitrary number of published counts. FPA approximates the result vector using a subset of its Discrete Fourier Transform (DFT) coefficients perturbed with Laplace noise. This may render the required noise scale in FPA smaller than that in LPA. However, the utility of the published result depends also on the quality of the DFT approximation. A small subset of DFT coefficients may effectively approximate time series with high amplitude in only a few frequencies (e.g., stock data). Nevertheless, in our application scenarios, the result vector may distribute its energy arbitrarily in the frequency domain. This could greatly impact the quality of the DFT approximation and, hence, the utility of FPA.

1.2 Our Contributions

Motivated by the shortcomings of LPA and FPA in the targeted problem, we introduce a novel mechanism that achieves ϵ-differential privacy with high utility via data preprocessing. Our technique decomposes the database columns into disjoint *groups*, and averages the counts in each group. Subsequently, it adds Laplace noise to each group average, and sets the result as the new count of every column in the group. Intuitively, the averaging *smooths* the counts in each group, acting as some form of preliminary perturbation. The actual purpose of this perturbation is to render the sensitivity on the new counts up to *constant* and, thus, *irrelevant* to the maximum number of the original counts affected by a user. As a result, the Laplace noise required for ϵ-differential privacy diminishes. We hereafter term our scheme as GS (for *grouping* and *smoothing*).

The challenging question that arises is, *how do we determine the grouping strategy*? Ideally, we must group together columns with *similar* counts, in order to reduce the smoothing perturbation. However, *optimally* grouping the original counts (e.g., via straightforward clustering) is a *careless* choice; we show that grouping compromises privacy when it operates on the original counts. On the other hand, grouping at *random* does not affect privacy, but may result in averaging counts with large variance, which increases the smoothing perturbation. To tackle this problem, we present an effective grouping technique with low privacy cost that is based on *sampling*. We initially adapt an existing sampling method, and explain its drawbacks. Subsequently, we design a novel sampling approach tailored to our setting. We analytically prove that our new scheme leads to better grouping than the existing one.

Furthermore, we observe that although larger groups substantially reduce the sensitivity of the new counts, they yield a stronger smoothing effect. For instance, if we choose only one group that contains all columns, the resulting average will practically destroy all the original counts, even though the final Laplace noise is negligible. On the other hand, we show that smaller groups require larger Laplace noise. *Fine-tuning* the group size seems a good direction for optimizing utility. However, it is not trivial how to do so without compromising privacy. Towards this end, we introduce a novel fine-tuning technique that maintains ϵ-differential privacy, while determining a group size that is close to optimal.

We provide a theoretical analysis of GS and a qualitative comparison with LPA and FPA. We explain that GS is expected to outperform both LPA and FPA in our targeted applications. The reason is that GS necessitates orders of magnitude smaller noise scale than LPA in scenarios with large sensitivity (e.g., "check-in" data). Moreover, the grouping and smoothing transformation of GS approximates the count vector much better than FPA, especially in settings where the counts have no natural ordering (such as in all our application examples).

Finally, we provide an exhaustive experimental evaluation using four real datasets, ranging from thousands to millions of users. The datasets come from various domains (e.g., the Gowalla geo-social network, the Netflix prize, etc.) and feature considerably different characteristics (e.g., number of columns, sensitivity, distribution, etc.). Our results confirm the superiority of GS over LPA and FPA, and its actual practicality in real settings.

The rest of the paper is organized as follows. Section 2 provides preliminary information and surveys the related work. Section 3 presents GS in detail. Section 4 analyzes the utility of GS and compares it with that of LPA and FPA. Section 5 includes our experiments. Finally, Section 6 concludes our work.

2. BACKGROUND

Section 2.1 presents the necessary preliminaries, whereas Section 2.2 surveys the related work.

2.1 Preliminaries

We present in turn our model, and the basic definitions and theorems revolving around ϵ-differential privacy.

Model A trusted curator holds a database D with user data (e.g., "check-ins"). We view D as a two-dimensional matrix, with n rows corresponding to users, and d columns representing attributes (e.g., places). The querier requests the *count* (i.e., number of non-zero cell values) of *every* column only *once*. For simplicity, we suppose that all cell values are *binary*, i.e., a non-zero cell is treated as 1 (e.g., a cell indicates whether a user visited a place or not). A row may have an *arbitrary* number of 1's. The requested counts are *non-overlapping*, i.e., each cell value affects *exactly one* count. Moreover, this is the *non-interactive* setting, where the curator handles these counts in a *batch* (in contrast to the *interactive* setting where the querier *adaptively* sends multiple queries in rounds). In essence, the non-interactive setting is equivalent to *one-time* publishing of private statistics about a database on a public web site. No more access is given to D by the curator, and the querier extracts any desired information (such as subsets of counts) directly from the published data.

ϵ-differential privacy Let $\mathcal{D}$ denote a set of finite databases with d attributes. Each $D \in \mathcal{D}$ is perceived as a set of rows. We make use of the following definition.

DEFINITION 1. *Two databases $D, D' \in \mathcal{D}$ are referred to as* **neighboring** *if*

$$|(D - D') \cup (D' - D)| = 1$$

Informally stated, D and D' are neighboring if we can obtain D' from D by removing or adding a single row.

A mechanism $\mathcal{M}$ is a randomized algorithm performed by the curator on a database, which applies some functionality

and outputs a *transcript* $\mathbf{t}$. The latter is returned to the querier in response to the query.

DEFINITION 2. *A mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{T}$ satisfies* **ϵ-differential privacy** *if for all sets $T \subseteq \mathcal{T}$, and every pair $D, D' \in \mathcal{D}$ of neighboring databases*

$$\Pr[\mathcal{M}(D) \in T] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(D') \in T]$$

The selection of ϵ is a social question [4]. Most typically, ϵ equals 0.1, $\ln 2$ or $\ln 3$. The smaller the value of ϵ, the stronger the privacy guarantees. Intuitively, $\mathcal{M}$ is considered to satisfy ϵ-differential privacy, if the participation of an individual in the database changes the probability distribution of $\mathcal{M}$ very slightly. This means that the adversary distinguishes with small probability whether a transcript $\mathbf{t}$ incorporates the data of an individual or not. Hence, the adversary infers no more information from $\mathbf{t}$ than the case where the individual's data were absent from the database.

Laplace Perturbation Algorithm (LPA [6, 4]) Before embarking on the details of LPA, we must formulate the *sensitivity* of a query w.r.t. $\mathcal{D}$. We model the query that asks for the count of each database column as a function $\mathbf{Q} : \mathcal{D} \to \mathbb{N}^d$, where d is the number of columns in the database. For $D, D' \in \mathcal{D}$, $\mathbf{Q}(D), \mathbf{Q}(D')$ are two d-dimensional vectors. We denote the L_p norm of $\mathbf{Q}(D), \mathbf{Q}(D')$ as $\|\mathbf{Q}(D) - \mathbf{Q}(D')\|_p$.

DEFINITION 3. *The L_p* **sensitivity** *of $\mathbf{Q}$ w.r.t. $\mathcal{D}$ is*

$$\Delta_p(\mathbf{Q}, \mathcal{D}) = \max_{D, D' \in \mathcal{D}} \left\|\mathbf{Q}(D) - \mathbf{Q}(D')\right\|_p$$

for all neighboring $D, D' \in \mathcal{D}$. When there is no ambiguity on $\mathcal{D}$, we simply use symbol $\Delta_p(\mathbf{Q})$.

Let $Lap(\lambda)$ be a random variable drawn from a Laplace distribution with mean zero and scale parameter λ. LPA achieves ϵ-differential privacy through the mechanism outlined in the following theorem.

THEOREM 1. *Let $\mathbf{Q} : \mathcal{D} \to \mathbb{N}^d$, and define $\mathbf{c} \stackrel{\text{def}}{=} \mathbf{Q}(D)$. A mechanism $\mathcal{M}$ that adds independently generated noise from a zero-mean Laplace distribution with scale parameter $\lambda = \Delta_1(\mathbf{Q})/\epsilon$ to each of the d output values of $\mathbf{Q}$, i.e., which produces transcript*

$$\mathbf{t} = \mathbf{c} + \langle Lap(\Delta_1(\mathbf{Q})/\epsilon)\rangle^d$$

enjoys ϵ-differential privacy. The error introduced in the i^{th} element of $\mathbf{t}$ by LPA is

$$error^i_{\mathsf{LPA}} = \mathbb{E}|\mathbf{t}[i] - \mathbf{c}[i]| = \mathbb{E}|Lap(\lambda)| = \sqrt{2}\lambda = \sqrt{2}\Delta_1(\mathbf{Q})/\epsilon$$

The proof is found in [4]. The higher the $\Delta_1(\mathbf{Q})$ or the smaller the ϵ, the larger the required Laplace noise for satisfying ϵ-differential privacy. Note that, in our setting, $1 \leq \Delta_1(\mathbf{Q}) \leq d$. For example, $\Delta_1(\mathbf{Q}) = 1$ if the columns represent histogram buckets, and each user affects only a *single* bucket count. Moreover, $\Delta_1(\mathbf{Q}) = d$ in time series data, where the user affects *all* counts. Finally, $\Delta_1(\mathbf{Q})$ may be any value in range $(1, d)$ controlled by the application *before* data collection. For instance, in geo-social networks, the maximum user "check-ins" (in *every* $D \in \mathcal{D}$) may be limited to a *fixed* value in the order of thousands, whereas d could be in the order of millions. We later explain that our mechanism features excellent performance for any $\Delta_1(\mathbf{Q}) > 1$.

Sampling Suppose we draw a random sample of rows from a database, and publish statistics about the sampled data instead. Sampling increases the uncertainty of the adversary about a user being in the sample. As a result, a lower Laplace noise is needed in order to satisfy ϵ-differential privacy for the sampled data. The following corollary is derived from Li et al. [11] for the ϵ-differential privacy framework.

COROLLARY 1. *Let $\mathcal{M}$ be a mechanism that satisfies ϵ_1-differential privacy. Let $\mathcal{M}_s$ be another mechanism that samples each row of its input with probability β, and then applies $\mathcal{M}$ on the resulting sample. $\mathcal{M}_s$ satisfies ϵ_2-differential privacy for $\epsilon_2 = \ln(1 + \beta(e^{\epsilon_1} - 1))$.*

The corollary essentially states that we achieve better privacy for the sample than for the original database, with the same amount of Laplace noise. Slightly restating it, we can maintain the same privacy level for the sample as that for the original database by injecting smaller Laplace noise.

2.2 Related Work

A plethora of differentially private techniques were introduced after the first proposal (LPA) in [6]. We categorize the existing methods based on their targeted setting, and identify our competitors.

Range-count queries Approaches of this category support ϵ-differential privacy in answering range-count queries, i.e., count queries with a range predicate (e.g., *"number of persons between 30 and 40 years old suffering from HIV"*). The original data are modeled as a *histogram*, where the sensitivity is *one* (i.e., each user affects a *single* count). The objective is to publish a perturbed version of the histogram, such that (i) arbitrary range-count queries can be answered, (ii) counts with longer ranges can be *directly* answered more accurately than aggregating a set of noisy counts with shorter ranges, and (iii) shorter ranges can be derived from larger ranges, taking advantage of *uniformity*.

Privelet [25] applies the Haar Wavelet Transform on the original histogram, and publishes a noisy version of its coefficient tree. The authors prove that higher nodes in the tree require smaller noise. Count queries with longer ranges can be directly answered using higher tree nodes. Xu et al. [27] propose two algorithms, *NoiseFirst* and *StructureFirst*. Both output a coarser and noisy version of an input histogram, which can directly answer counts with longer ranges more accurately. *NoiseFirst* initially applies noise, and then merges consecutive histogram buckets. *StructureFirst* initially creates the coarser histogram by merging *adjacent* buckets, and then injects noise to the summation of their counts. Xiao et al. [26] address 2-dimensional range-count queries. They merge *adjacent* buckets using a kd-tree partitioning technique, and answer shorter range-counts from larger ones. The shorter range-counts are more accurate, because the Laplace noise they incorporate constitutes only a portion of the noise injected to the larger range-count.

Query consistency Hay et al. [9] consider a setting where the published statistics follow certain consistency constraints (e.g., counts that are in sorted order, or counts with some linear relationship). They focus on histograms and range-count queries. They apply a data transformation technique in conjunction with noise injection, in order to provide ϵ-differential privacy, while satisfying the given constraints.

Sparse data Cormode et al. [2] aim at publishing a summary of a contingency table with ϵ-differential privacy. They observe that, when the contingency table is sparse, considerable computational savings can be achieved during the summary generation. Specifically, instead of producing the summary *after* perturbing the entire contingency table (which is an expensive process), they derive it *directly* from the original data. The computational cost is proportional to the summary size (which is controlled by the owner), and the utility is similar to that of the noisy contingency table. Li et al. [12] release a perturbed version of the *entire* database, instead of specific noisy statistics. The output can be used for answering an arbitrary number of queries, without extra privacy cost. Their mechanism is based on *compressive sensing*, which constructs a synopsis of the initial data. Subsequently, it injects noise to this synopsis, and uses the result to re-construct a perturbed database. In order for this mechanism to be effective, the original data must be sparse.

Correlated queries Schemes in this group exploit correlations among *overlapping* queries, in order to reduce the amount of noise required for ϵ-differential privacy. *K-norm* [8] adjusts the noise by analyzing the queries using concepts from convex geometry. Li et al. [10] proposed a two-stage matrix mechanism, which has been recently subsumed by Yuan et al. [28]. The latter argue that the matrix mechanism features considerable computational overhead and suboptimal strategies. They introduce an improved mechanism (in terms of both accuracy and computation), which is based on the theory of *low-rank approximation*.

Adaptive queries In the *interactive setting*, the queries arrive adaptively over time at the curator. The latter answers them immediately with no knowledge of future queries. LPA is adapted to this setting by assuming an a priori privacy budget; it keeps on independently perturbing every answer until the budget is depleted, and then shuts down. An alternative adaptation works for an arbitrary number of queries, but doubles the required noise every time a query arrives. Roth and Roughgarden [19] propose the *median mechanism*, which improves upon LPA by answering some of the queries using the noisy answers of previous ones.

Minimization of relative error Contrary to the aforementioned mechanisms that mainly focus on reducing the absolute error in the published data, *iReduct* [23] targets at minimizing the relative error. It achieves this by calibrating the Laplace noise scale proportionally to the magnitude of the query answers. Specifically, it injects low noise into answers with small values, shifting a larger amount of noise to larger outputs.

Non-numerical query answers There are scenarios where the query outputs are *nominal*. Adding numerical noise to such answers does not make sense. The *exponential mechanism* [15] targets at such settings. It employs a utility function that measures the quality of each possible output. It then chooses the answer to be published among all possible outputs, with probability that increases exponentially with utility. This leads to ϵ-differential privacy.

Variations of ϵ-differential privacy There exist two relaxations of ϵ-differential privacy; (ϵ, δ)-*probabilistic differential privacy* [13] and (ϵ, δ)-*indistinguishability* [5, 16]. The former achieves ϵ-differential privacy with high probability ($\geq 1 - \delta$), whereas the latter relaxes the bound of ϵ-differential privacy in Definition 2. Götz et al. [7] prove that (ϵ, δ)-probabilistic differential privacy is a stronger notion than (ϵ, δ)-indistinguishability.

Our competitors Our goal is to minimize the *absolute* error when publishing *numerical* answers of *non-overlapping* count queries in the *non-interactive* setting with ϵ-*differential privacy*. Therefore, the methods of the last five categories above are orthogonal to our work. Moreover, we make no assumptions about the original data or the query answers and, hence, techniques requiring sparse data or query consistency are not applicable to our setting. On the other hand, [25, 26, 27] optimize range-count queries on histograms, where a user affects only a *single* count. In contrast, we focus on (non-range) *count* queries where a user affects an *arbitrary* number of answers.

We identify two competitors. The first is a straightforward application of LPA [6]. The second is an *adaptation* of the work by Rastogi and Nath [18] to our setting. Specifically, we focus on the *centralized* mechanism of [18], called FPA, which assumes a database D where the rows are *time-series* (e.g., sequences of periodic reports on the health condition of a patient), and the columns are timestamps. It publishes with ϵ-differential privacy the count for every timestamp. The authors consider that a user affects *all* counts, but here we adapt it such that the user may affect an *arbitrary* number of counts. FPA first applies the *Discrete Fourier Transform* (DFT) on the count vector $\mathbf{Q}(D)$. Next, it chooses the first ρ ($\ll d$) DFT coefficients, and injects Laplace noise of certain scale to them. Finally, it runs the inverse DFT on the noisy coefficients and publishes the result. The following theorem states the required noise scale in FPA for achieving ϵ-differential privacy, as well as its resulting error.

THEOREM 2. *Let d be the number of columns in D. In order to satisfy ϵ-differential privacy,* FPA *mandates injecting Laplace noise with scale $\lambda = \frac{\sqrt{\rho \cdot d \cdot \Delta_1(\mathbf{Q})}}{\epsilon}$ to each of the ρ selected DFT coefficients. Moreover, let $RE^i_{\mathsf{FPA}}(\rho)$ denote the re-construction error of count i due to the DFT approximation. Then, the total error of* FPA *for count i is $error^i_{\mathsf{FPA}} \leq RE^i_{\mathsf{FPA}}(\rho) + \sqrt{2} \cdot \frac{\rho \cdot \sqrt{\Delta_1(\mathbf{Q})}}{\epsilon \cdot \sqrt{d}}$.*

PROOF. See Appendix A. □

In the sequel, we introduce our novel mechanism, and provide a utility analysis and experimental comparison with LPA and FPA.

3. GROUPING AND SMOOTHING

In this section we present our proposed scheme, called GS, which is based on grouping and smoothing. In order to better demonstrate its underlying ideas, we construct GS gradually by starting with simple components and later adding more advanced features. Specifically, in Section 3.1 we explain a simple mechanism, referred to as GS-R, which relies on random grouping. In Section 3.2, we design an improvement of GS-R, called GS-S, which implements a more sophisticated grouping technique through sampling. Finally, in Section 3.3 we augment GS-S with a fine-tuning step to derive our full-fledged GS mechanism.

Table 1: Summary of notation

Symbol	Meaning
D / $\mathcal{D}$	Input database / Set of all databases
D_s / $\mathcal{D}_s$	Sampled database / Set of all sampled databases
$\mathcal{G}$ / $\mathbf{g}$	Grouping comput. module / Output strategy of $\mathcal{G}$
w / $\mathbf{g}_w$	Group size / Strategy with w-sized groups
d / $d_{\mathbf{g}}$	Number of columns in D / Cardinality of $\mathbf{g}$
$\mathbf{Q}$	Query that asks for all column counts
$\mathbf{c}$ / $\mathbf{c}_s$	Result vector of $\mathbf{Q}$ on D / D_s
$\mathbf{Q_g}$ / $\mathbf{c_g}$	Count query given $\mathbf{g}$ / Result vector of $\mathbf{Q_g}$ on D
$\mathcal{M}_{\mathbf{g}}$ / $\mathbf{t_g}$	Mechanism that perturbs $\mathbf{c_g}$ / Output of $\mathcal{M}_{\mathbf{g}}$
$\mathcal{M}_s$ / $\mathbf{t}_s$	Sampling sub-module of $\mathcal{G}$ / Output of $\mathcal{M}_s$
$\mathbf{t}$	Final output of mechanism GS-R, GS-S or GS
$\Delta_p(\mathbf{Q})$	L_p sensitivity of $\mathbf{Q}$ w.r.t. $\mathcal{D}$

A vital concept common in all three mechanisms is the *grouping strategy*. Therefore, we provide its definition here.

DEFINITION 4. *Let $D \in \mathcal{D}$ be a database. A* **grouping strategy** *is a* partition *of the columns of D. It is viewed as a vector* $\mathbf{g}$*, whose elements are sets (called* groups*) of column identifiers. We refer to the computational module that produces* $\mathbf{g}$ *as* $\mathcal{G}$*.*

The grouping strategy dictates the smoothing magnitude, as well as the amount of added noise. Essentially, what mainly differentiates GS-R, GS-S and GS is the construction of their grouping strategy, i.e., module $\mathcal{G}$. Table 1 summarizes the most important notation.

3.1 Random Grouping (GS-R)

Figure 1 sketches the GS-R mechanism, which takes as inputs a database D, the number of columns d of D, the query $\mathbf{Q}$, and its sensitivity $\Delta_1(\mathbf{Q})$. We uniquely identify each column by a number in $\{1, 2, \ldots, d\}$. GS-R initially generates a grouping strategy $\mathbf{g}$ via module $\mathcal{G}$. Specifically, it *randomly* partitions the columns of D into $d_{\mathbf{g}} = \lfloor d/\Delta_1(\mathbf{Q}) \rfloor$ distinct groups. The j^{th} group is denoted by $\mathbf{g}[j]$ and contains exactly $\Delta_1(\mathbf{Q})$ identifiers, except for the last group $\mathbf{g}[d_{\mathbf{g}}]$ that contains $\Delta_1(\mathbf{Q}) + (d \bmod \Delta_1(\mathbf{Q}))$ identifiers.

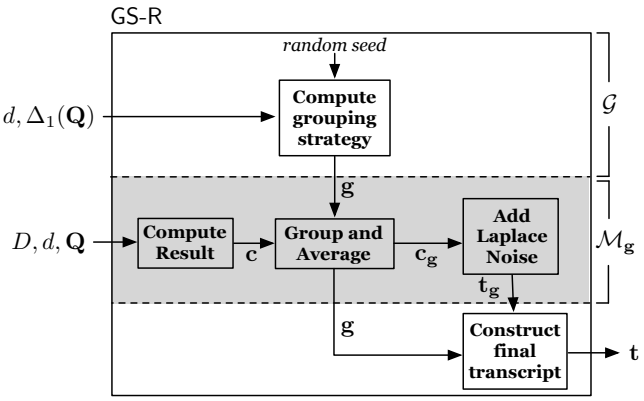


Figure 1: Outline of GS-R

Subsequently, GS-R runs module $\mathcal{M}_{\mathbf{g}}$. The latter computes the answer vector $\mathbf{c} = \mathbf{Q}(D)$. Then, it generates a vector $\mathbf{c_g}$ of size $d_{\mathbf{g}}$, where the j^{th} element corresponds to $\mathbf{g}[j]$ and holds the *average* of the counts of the columns in $\mathbf{g}[j]$. Next, it adds independent Laplace noise to each element of $\mathbf{c_g}$, producing vector $\mathbf{t_g}$. In GS-R, this noise must be

Input: $D, d, \mathbf{Q}, \Delta_1(\mathbf{Q})$
Output: $\mathbf{t}$

/* *Module* $\mathcal{G}$ */
1. Set $d_{\mathbf{g}} = \lfloor d/\Delta_1(\mathbf{Q}) \rfloor$
2. Initialize $I = \langle 1, 2, \ldots, d \rangle$ and empty vector $\mathbf{g}$ of size $d_{\mathbf{g}}$
3. Randomly permute I
4. For $j = 1$ to $(d_{\mathbf{g}} - 1)$
5. $\quad \mathbf{g}[j] = \{I[(j-1) * \Delta_1(\mathbf{Q}) + 1], \ldots, I[j * \Delta_1(\mathbf{Q})]\}$
6. $\mathbf{g}[d_{\mathbf{g}}] = \{I[(d_{\mathbf{g}} - 1) * \Delta_1(\mathbf{Q}) + 1], \ldots, I[d]\}$

/* *Module* $\mathcal{M}_g$ */
7. Initialize $\mathbf{c} = \mathbf{Q}(D)$ and empty vector $\mathbf{c_g}$ of size $d_{\mathbf{g}}$
8. For $j = 1$ to $d_{\mathbf{g}}$
9. $\quad \mathbf{c_g}[j]$ is the average of every $\mathbf{c}[i]$ such that $i \in \mathbf{g}[j]$
10. Compute $\mathbf{t_g} = \mathbf{c_g} + \langle Lap(1/\epsilon) \rangle^{d_{\mathbf{g}}}$

11. Initialize empty vector $\mathbf{t}$ of size d
12. For $i = 1$ to d
13. $\quad \mathbf{t}[i] = \mathbf{t_g}[j]$, such that $i \in \mathbf{g}[j]$

Figure 2: Pseudocode of GS-R

$Lap(1/\epsilon)$. Eventually, it constructs $\mathbf{t}$, which stores the final output. The count of column i in $\mathbf{t}$ is assigned the value from $\mathbf{t_g}$ that corresponds to group $\mathbf{g}[j]$ containing i. Figure 2 provides the pseudocode of GS-R.

We next show that GS-R satisfies ϵ-differential privacy. Before embarking on the detailed proof, in order to facilitate presentation, we define query $\mathbf{Q_g}$ that depends on a *given* grouping strategy $\mathbf{g}$.

DEFINITION 5. *Let $\mathbf{g}$ be a grouping strategy with $d_{\mathbf{g}}$ groups. We define query $\mathbf{Q_g} : \mathcal{D} \to \mathbb{R}^{d_{\mathbf{g}}}$. Given D as an input, $\mathbf{Q_g}$ outputs a vector of $d_{\mathbf{g}}$ elements, where the j^{th} element is the average of the counts of D's columns in $\mathbf{g}[j]$.*

Observe that $\mathbf{c_g} = \mathbf{Q_g}(D)$ in Figures 1 and 2. Considering the construction of $\mathbf{g}$ in GS-R, the following lemma explains the sensitivity of $\mathbf{Q_g}$ in this mechanism.

LEMMA 1. *In GS-R, it holds that $\Delta_1(\mathbf{Q_g}) \leq 1$.*

PROOF. Let u be a user in D. Denote by $m_{u,j}$ the number of 1's that u has in the columns belonging to group $\mathbf{g}[j]$. Also recall that u has at most $\Delta_1(\mathbf{Q})$ 1's in total, i.e., $\sum_{j=1}^{d_{\mathbf{g}}} m_{u,j} \leq \Delta_1(\mathbf{Q})$. Suppose that we remove u from D to derive neighboring database D'. Observe that u affects element j of $\mathbf{Q_g}(D)$ by *at most* $m_{u,j}/\Delta_1(\mathbf{Q})$. Using Definition 3, it follows that

$$\Delta_1(\mathbf{Q_g}) = \max_{D, D' \in \mathcal{D}} \sum_{j=1}^{d_{\mathbf{g}}} \left| \mathbf{Q_g}(D)_j - \mathbf{Q_g}(D')_j \right|$$

$$\leq \max_{D \in \mathcal{D} \wedge u \in D} \sum_{j=1}^{d_{\mathbf{g}}} \frac{m_{u,j}}{\Delta_1(\mathbf{Q})} = \frac{\Delta_1(\mathbf{Q})}{\Delta_1(\mathbf{Q})} = 1$$

which concludes our proof. □

We are now ready to establish the privacy of GS-R.

THEOREM 3. *GS-R satisfies ϵ-differential privacy.*

PROOF. Let $\mathcal{M} : \mathcal{D} \to \mathcal{T}$ denote the GS-R mechanism. We need to prove that $\mathcal{M}$ satisfies Definition 2. We assume that, in the worst case, the adversary learns the followed

strategy $\mathbf{g}$ from the output. Therefore, in the view of the adversary, it holds for any $\mathbf{t} \in \mathcal{T}$ that

$$\begin{aligned}\Pr[\mathcal{M}(D) = \mathbf{t}] &= \Pr[\mathcal{M}(D) = \mathbf{t} \wedge \mathcal{G}(D) = \mathbf{g}] \\ &= \Pr[\mathcal{M}(D) = \mathbf{t} | \mathcal{G}(D) = \mathbf{g}] \cdot \Pr[\mathcal{G}(D) = \mathbf{g}]\end{aligned} \quad (1)$$

When $\mathcal{M}$ produces $\mathbf{t}$ *given* a grouping strategy $\mathbf{g}$, it becomes equivalent to sub mechanism $\mathcal{M}_{\mathbf{g}}$. Formula 1 becomes

$$\Pr[\mathcal{M}(D) = \mathbf{t}] = \Pr[\mathcal{M}_{\mathbf{g}}(D) = \mathbf{t}_{\mathbf{g}}] \cdot \Pr[\mathcal{G}(D) = \mathbf{g}] \quad (2)$$

Moreover, since the sensitivity of $\mathbf{Q}_{\mathbf{g}}$ is at most 1 (Lemma 1) and $\mathcal{M}_{\mathbf{g}}$ injects Laplace noise with scale $1/\epsilon$, the following holds combining Theorem 1 and Definition 2:

$$\Pr[\mathcal{M}_{\mathbf{g}}(D) = \mathbf{t}_{\mathbf{g}}] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}_{\mathbf{g}}(D') = \mathbf{t}_{\mathbf{g}}] \quad (3)$$

In addition, $\mathbf{g}$ in GS-R is computed randomly and independently of the input database. Therefore,

$$\Pr[\mathcal{G}(D) = \mathbf{g}] = \Pr[\mathcal{G}(D') = \mathbf{g}] \quad (4)$$

Finally, combining Formulae 2-4 we get

$$\begin{aligned}\Pr[\mathcal{M}(D) = \mathbf{t}] &\leq e^{\epsilon} \cdot \Pr[\mathcal{M}_{\mathbf{g}}(D') = \mathbf{t}_{\mathbf{g}}] \cdot \Pr[\mathcal{G}(D') = \mathbf{g}] \\ &= e^{\epsilon} \cdot \Pr[\mathcal{M}(D') = \mathbf{t}]\end{aligned} \quad (5)$$

The above holds for every $\mathbf{t} \in \mathcal{T}$. Hence, for any $T \subseteq \mathcal{T}$,

$$\begin{aligned}\Pr[\mathcal{M}(D) \in T] &= \sum_{\mathbf{t} \in T} \Pr[\mathcal{M}(D) = \mathbf{t}] \\ &\leq \sum_{\mathbf{t} \in T} e^{\epsilon} \cdot \Pr[\mathcal{M}(D') = \mathbf{t}] \\ &= e^{\epsilon} \cdot \Pr[\mathcal{M}(D') \in T]\end{aligned} \quad (6)$$

which concludes our proof. □

To sum up, due to grouping and smoothing, GS-R necessitates low Laplace noise for guaranteeing ϵ-differential privacy. However, smoothing inflicts extra perturbation error. If the grouping strategy is not designed carefully, columns with considerably different counts may be grouped together. As a result, their smoothed counts (prior to Laplace noise injection) will greatly deviate from their original ones. Since GS-R *randomly* selects the groups, "bad" grouping strategies will often occur. We next describe GS-S, which tackles this problem via an effective grouping technique.

3.2 Effective Grouping via Sampling (GS-S)

In order to achieve grouping that minimizes the perturbation due to smoothing, we should group together columns with *similar* counts. A straightforward solution is to *optimally* group the columns, by sorting them based on their *original* counts, and then partitioning them into $d_{\mathbf{g}}$ consecutive groups. Unfortunately, this violates ϵ-differential privacy, because the optimal grouping for D may be *different* from that for neighboring database D' with *substantial* probability. We demonstrate this below with an example.

Consider a database D with columns $\{1, 2, 3, 4\}$, $\Delta_1(\mathbf{Q}) = 2$, and column counts $\mathbf{c} = \langle 11, 13, 13, 14 \rangle$. Assume that we derive a neighboring database D', by removing from D a user with data $\langle 0, 0, 1, 0 \rangle$. The column counts of D' are $\mathbf{c}' = \langle 11, 13, 12, 14 \rangle$. Let $\mathcal{G}$ be a grouping procedure that optimally groups the columns. Then, $\mathcal{G}(D)$ returns $\mathbf{g} = \langle \{1, 2\}, \{3, 4\} \rangle$ with probability 1. On the other hand, $\mathcal{G}(D')$ outputs $\mathbf{g}' = \langle \{1, 3\}, \{2, 4\} \rangle$ with probability 1, which means that it produces $\mathbf{g}$ with probability 0.

Suppose that we substitute the strategy computation module $\mathcal{G}$ of GS-R with the optimal technique described above. In order to prove that the modified mechanism satisfies ϵ-differential privacy, we follow the same steps as in the proof of Theorem 3 until Formula 3. However, we just showed above that Formula 4 does not hold, and also that *there exists at least one* $\mathbf{t} = \mathcal{M}(D)$, such that

$$\Pr[\mathcal{M}(D) = \mathbf{t}] \not\leq e^{\epsilon} \cdot \Pr[\mathcal{M}_{\mathbf{g}}(D') = \mathbf{t}_{\mathbf{g}}] \cdot \Pr[\mathcal{G}(D') = \mathbf{g}] = 0$$

for *any* ϵ. Consequently, Formula 6 does not hold and ϵ-differential privacy is violated. The above suggests that whenever a module "looks into" the original data D (such as $\mathcal{G}$ above), noise must be injected to derive a private result. However, injecting noise to $\mathbf{c} = \mathbf{Q}(D)$ is prohibitive, due to the potentially large $\Delta_1(\mathbf{Q})$.

GS-S tackles this challenge by creating the optimal strategy for a *sample* D_s of D, instead for D itself. The key idea is that $\mathbf{c}_s = \mathbf{Q}(D_s)$ necessitates *much lower* noise than $\mathbf{c}$. Therefore, the noisy results are *representative* enough to yield a strategy close to the actual optimal w.r.t. D.

Figure 3 outlines the steps of GS-S, which share similarities with that of GS-R. In fact, GS-S features only two differences compared to GS-R: (i) $\mathcal{G}$ devises the grouping strategy $\mathbf{g}$ in a fundamentally different manner, and (ii) after computing $\mathbf{c}_{\mathbf{g}}$ based on $\mathbf{c}$ and $\mathbf{g}$, $\mathcal{M}_{\mathbf{g}}$ injects noise $Lap(2/\epsilon)$ to derive $\mathbf{t}_{\mathbf{g}}$, whereas in GS-R it applies noise $Lap(1/\epsilon)$.

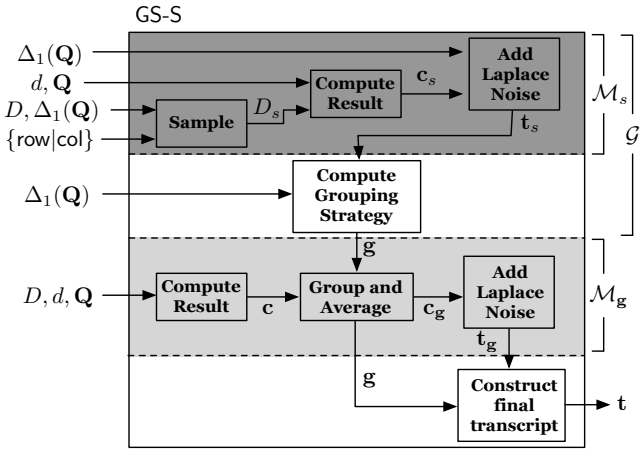


Figure 3: Outline of GS-S

We next explain how the $\mathcal{G}$ module of GS-S generates $\mathbf{g}$. It first samples D, creating a small portion of the original database. This is denoted by D_s and has the same schema as D. The procedure for deriving D_s from D is clarified soon. Subsequently, it computes the column counts on D_s, represented by vector $\mathbf{c}_s$. Next, it injects Laplace noise into $\mathbf{c}_s$ and generates perturbed transcript $\mathbf{t}_s$. Finally, it computes the optimal grouping strategy (i.e., via sorting) w.r.t. $\mathbf{t}_s$. Observe in Figure 3 that $\mathcal{G}$ practically integrates a private mechanism $\mathcal{M}_s$ (dark grey area on top) that produces $\mathbf{t}_s$ as the noisy version of $\mathbf{c}_s = \mathbf{Q}(D_s)$. Figure 4 includes the pseudocode of GS-S.

We present in turn two sampling techniques for deriving D_s from D. The first samples rows from D and relies on results from [11]. The second is a novel approach that samples column values (i.e., 1's) from rows. For each technique, we explain the Laplace noise required for $\mathbf{c}_s$, and prove that GS-S achieves ϵ-differential privacy when employing it.

Input: $D, d, \mathbf{Q}, \Delta_1(\mathbf{Q})$, sampling_method
Output: $\mathbf{t}$

/* *Module* $\mathcal{G}$ */
  /* *Start of submodule* $\mathcal{M}_s$ *in* $\mathcal{G}$ */
1. If sampling_method = row
2. Get D_s by sampling rows from D with rate $\frac{e^{\epsilon/2}-1}{e^{\epsilon \cdot \Delta_1(\mathbf{Q})/2}-1}$
3. Else if sampling_method = col
4. Get D_s by sampling a single 1 from each row of D
5. $\mathbf{c}_s = \mathbf{Q}(D_s)$ and $\mathbf{t}_s = \mathbf{c}_s + \langle Lap(2/\epsilon)\rangle^d$
  /* *End of submodule* $\mathcal{M}_s$ *in* $\mathcal{G}$ */
6. Same as Lines 1-2 in Figure 2
7. Sort I based on the corresponding counts in $\mathbf{t}_s$
8. Same as Lines 4-6 in Figure 2

/* *Module* $\mathcal{M}_g$ */
9. Same as Lines 7-9 in Figure 2
10. Compute $\mathbf{t_g} = \mathbf{c_g} + \langle Lap(2/\epsilon)\rangle^{d_\mathbf{g}}$

11. Same as Lines 11-13 in Figure 2

Figure 4: Pseudocode of GS-S

Row Sampling In order to generate D_s, this technique first samples the rows of D with rate $\beta = \frac{e^{\epsilon/2}-1}{e^{\epsilon \cdot \Delta_1(\mathbf{Q})/2}-1}$. Next, it derives $\mathbf{c}_s$ on D_s, and injects noise $Lap(2/\epsilon)$ in order to produce transcript $\mathbf{t}_s$. This particular choice of β and noise scale is necessary for guaranteeing ϵ-differential privacy in GS-S with row sampling.

THEOREM 4. GS-S *integrating row sampling satisfies* ϵ*-differential privacy.*

PROOF. Let $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{T}$ be the GS-S mechanism integrating the row sampling technique. Suppose that $\mathcal{G}$ is the module that calculates the grouping strategy $\mathbf{g}$, and $\mathbf{Q_g}$ the query described in Definition 5. Let $\mathcal{M_g}$ be the mechanism that returns $\mathbf{t_g}$, which is the result of $\mathbf{Q_g}$ perturbed with noise $Lap(2/\epsilon)$. Following a similar rationale as in the proof of Theorem 3, it holds that

$$\Pr[\mathcal{M}(D) = \mathbf{t}] = \Pr[\mathcal{M_g}(D) = \mathbf{t_g}] \cdot \Pr[\mathcal{G}(D) = \mathbf{g}] \quad (7)$$

i.e., the probability to get transcript $\mathbf{t}$ from $\mathcal{M}$ depends on the mechanism $\mathcal{M_g}$ that perturbs the already grouped and averaged counts, *and* the module $\mathcal{G}$ that computes the specific grouping strategy $\mathbf{g}$ followed by $\mathcal{M_g}$.

Since $\Delta_1(\mathbf{Q_g}) \leq 1$ (from Lemma 1) and $\mathcal{M_g}$ injects noise $Lap(2/\epsilon)$, $\mathcal{M_g}$ satisfies $\frac{\epsilon}{2}$-differential privacy according to Theorem 1. Consequently, it holds

$$\Pr[\mathcal{M_g}(D) = \mathbf{t_g}] \leq e^{\frac{\epsilon}{2}} \cdot \Pr[\mathcal{M_g}(D') = \mathbf{t_g}] \quad (8)$$

Let $T_\mathbf{g} \subseteq \mathcal{T}$ be the set of *all* transcripts $\mathbf{t}_s$ that yield strategy $\mathbf{g}$ in GS-S. Also recall that $\mathcal{M}_s$ is the mechanism that gets as input dataset D, samples the rows of D with rate $\beta = \frac{e^{\epsilon/2}-1}{e^{\epsilon \cdot \Delta_1(\mathbf{Q})/2}-1}$ retrieving D_s, and produces $\mathbf{t}_s$ on D_s. Since $\mathcal{G}$ eventually derives $\mathbf{g}$ based solely on $\mathbf{t}_s$, it holds

$$\Pr[\mathcal{G}(D) = \mathbf{g}] = \Pr[\mathcal{M}_s(D) \in T_\mathbf{g}] \quad (9)$$

Transcript $\mathbf{t}_s$ is the answer $\mathbf{Q}(D_s) = \mathbf{c}_s$ perturbed with noise $Lap(2/\epsilon)$. If $\mathcal{M}_s$ does not perform sampling, then it is equivalent to a mechanism $\mathcal{M}'$ that outputs answer $\mathbf{Q}(D)$ perturbed with noise $Lap(2/\epsilon)$. Thus, $\mathcal{M}'$ enjoys ($\epsilon_1 = \frac{\epsilon \cdot \Delta_1(\mathbf{Q})}{2}$)-differential privacy based on Theorem 1. Due to Corollary 1, $\mathcal{M}_s$ achieves ϵ_2-differential privacy, where $\epsilon_2 = \ln(1 + \beta(e^{\epsilon_1} - 1)) = \ln(1 + \frac{e^{\epsilon/2}-1}{e^{\epsilon \cdot \Delta_1(\mathbf{Q})/2}-1}(e^{\epsilon \cdot \Delta_1(\mathbf{Q})/2} - 1)) = \frac{\epsilon}{2}$. It follows that

$$\Pr[\mathcal{M}_s(D) \in T_\mathbf{g}] \leq e^{\frac{\epsilon}{2}} \cdot \Pr[\mathcal{M}_s(D') \in T_\mathbf{g}] \quad (10)$$

Combining Formulae 9 and 10, we get

$$\Pr[\mathcal{G}(D) = \mathbf{g}] \leq e^{\frac{\epsilon}{2}} \cdot \Pr[\mathcal{G}(D') = \mathbf{g}] \quad (11)$$

Using Formulae 7, 8 and 11 we derive

$$\begin{aligned}\Pr[\mathcal{M}(D) = \mathbf{t}] &\leq e^{\frac{\epsilon}{2}} \cdot \Pr[\mathcal{M_g}(D') = \mathbf{t_g}] \cdot e^{\frac{\epsilon}{2}} \cdot \Pr[\mathcal{G}(D') = \mathbf{g}] \\ &= e^{\epsilon} \cdot \Pr[\mathcal{M}(D') = \mathbf{t}] \quad (12)\end{aligned}$$

Adding the probabilities over any $T \subseteq \mathcal{T}$ (similar to the proof of Theorem 3) yields ϵ-differential privacy. □

If sampling is not used at all, $\mathbf{c}_s$ becomes equal to $\mathbf{c}$, and must receive $Lap(2 \cdot \Delta_1(\mathbf{Q})/\epsilon)$ so that GS-S retains ϵ-differential privacy. Row sampling enables GS-S to reduce this noise to $Lap(2/\epsilon)$. Nevertheless, its main drawback is that sampling rate β decreases *exponentially* with the sensitivity, potentially yielding a very small sample. Therefore, the resulting $\mathbf{c}_s$ may not be representative of $\mathbf{c}$. This fact may considerably balance off the benefits from Laplace noise reduction, sometimes even completely negating them. The column sampling technique presented below tackles this problem.

Column Sampling This method keeps *all* the rows of D in the sample D_s. However, it preserves only a *single* 1 per row (i.e., it samples column information). Specifically, it randomly selects one 1 per row, setting the remaining column values to 0. Subsequently, the technique derives $\mathbf{c}_s$ from D_s, and injects noise $Lap(2/\epsilon)$ to it to produce $\mathbf{t}_s$. Column sampling has a similar effect to the case of row sampling; the noise required on $\mathbf{c}_s$ for achieving ϵ-differential privacy becomes lower than the case where no sampling is used at all.

THEOREM 5. GS-S *integrating column sampling satisfies* ϵ*-differential privacy.*

PROOF. Let $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{T}$ be the GS-S mechanism integrating the column sampling technique. Throughout this proof, we use explicit notation $\Delta_1(\mathbf{Q}, \mathcal{D})$ instead of $\Delta_1(\mathbf{Q})$. Suppose that $\mathcal{G}$ is the module that calculates the grouping strategy $\mathbf{g}$, and $\mathbf{Q_g}$ the query described in Definition 5. Let $\mathcal{M_g}$ be the mechanism that returns $\mathbf{t_g}$, which is the result of $\mathbf{Q_g}$ perturbed with noise $Lap(2/\epsilon)$. Moreover, let $T_\mathbf{g} \subseteq \mathcal{T}$ be the set of all transcripts $\mathbf{t}_s$ that yield strategy $\mathbf{g}$ in GS-S. Also let $\mathcal{M}_s$ be the mechanism that gets dataset D as input, samples D_s from D by maintaining *exactly* one random 1 in each row of D, and produces $\mathbf{t}_s$ on D_s. Transcript $\mathbf{t}_s$ is answer $\mathbf{Q}(D_s) = \mathbf{c}_s$ perturbed with noise $Lap(2/\epsilon)$.

Following an identical rationale as in the proof of Theorem 4, Formulae 7-9 hold. The only difference between GS-S with row sampling and GS-S with column sampling is mechanism $\mathcal{M}_s$. In Theorem 4 we proved that the $\mathcal{M}_s$ with row sampling achieves $\frac{\epsilon}{2}$-differential privacy, which further allowed us to prove Formulae 10-12. Consequently, in order to complete this proof, it suffices to prove that the $\mathcal{M}_s$ with column sampling also achieves $\frac{\epsilon}{2}$-differential privacy. In the sequel, we focus on this task.

We start by analyzing the set of *all possible* D_s derived from sampling *any* $D \in \mathcal{D}$, henceforth denoted by $\mathcal{D}_s$. Due

to the column sampling technique, every row in *any* $D_s \in \mathcal{D}_s$ contains exactly a *single* 1. Thus, removing a row from D_s affects $\mathbf{c}_s$ by exactly one. This means that the sensitivity of $\mathbf{Q}$ w.r.t. $\mathcal{D}_s$ is $\Delta_1(\mathbf{Q}, \mathcal{D}_s) = 1$. Recall that $\mathcal{M}_s$ computes $\mathbf{Q}(D_s)$, instead of $\mathbf{Q}(D)$. Therefore, the sensitivity in $\mathcal{M}_s$ is $\Delta_1(\mathbf{Q}, \mathcal{D}_s)$, and *not* $\Delta_1(\mathbf{Q}, \mathcal{D})$. Combining the above with the fact that $\mathcal{M}_s$ injects noise $Lap(2/\epsilon)$, we derive that $\mathcal{M}_s$ achieves $\frac{\epsilon}{2}$-differential privacy based on Theorem 1. This concludes our proof. □

Both row and column sampling inject the same amount of Laplace noise to $\mathbf{c}_s$. However, the sample size in column sampling has a *linear* dependence on $\Delta_1(\mathbf{Q})$. The reason is that a row may have up to $\Delta_1(\mathbf{Q})$ 1's and, hence, the probability of preserving any 1 from a row of D in D_s is at least $1/\Delta_1(\mathbf{Q})$. Consequently, D_s leads to a more representative $\mathbf{c}_s$ in column sampling than in its row counterpart, resulting in a better strategy $\mathbf{g}$. In Section 4, we analytically prove the superiority of column sampling over row sampling.

3.3 Adding a Fine-tuning Step (GS)

Recall that the grouping strategy in GS-S assumes that the group size is equal to $\Delta_1(\mathbf{Q})$. This choice renders the sensitivity of $\mathbf{Q}_\mathbf{g}$ at most 1 and, thus, the $\mathcal{M}_\mathbf{g}$ sub mechanism mandates Laplace noise with scale $2/\epsilon$. However, the extra perturbation induced by smoothing is gravely affected by the group size. In databases where $\Delta_1(\mathbf{Q})$ is large, this perturbation may be strong. The idea that arises is to reduce the group size, in order to alleviate the effect of smoothing. Nevertheless, this would negate the proof of Lemma 1, i.e., it would no longer hold that $\Delta_1(\mathbf{Q}_\mathbf{g}) \leq 1$. In fact, we will show that the sensitivity increases as the group size decreases. As a consequence, the Laplace noise in $\mathcal{M}_\mathbf{g}$ increases as well.

Motivated by the above discussion, we suggest *tuning* the group size in order to find the best *trade-off* between the perturbation of smoothing and that of the required Laplace noise in $\mathcal{M}_\mathbf{g}$. Finding the optimal group size is not trivial; we need to evaluate each group size in terms of utility, without compromising privacy. However, assessing utility by "looking into" the initial data (either D or D_s) violates ϵ-differential privacy. Specifically, similar to the example in Section 3.2, the optimal group size for D may be different from that for D' with substantial probability.

In this section we address the above challenge. In particular, we augment the $\mathcal{G}$ module of GS-S with a fine-tuning step for the group size, which is based on useful *estimates* about the final error from the private transcript $\mathbf{t}_s$ returned by sub mechanism $\mathcal{M}_s$. This step outputs a grouping strategy that is close to the optimal, without introducing any additional privacy cost.

Figure 5 outlines our full-fledged GS mechanism, whereas Figure 6 provides the GS pseudocode. The $\mathcal{M}_s$ sub mechanism of GS is identical to that in GS-S. The $\mathcal{M}_\mathbf{g}$ sub mechanism of GS is similar to that in GS-S, with the difference that it adds $Lap(\frac{2\cdot\Delta_1(\mathbf{Q})}{\epsilon\cdot w})$, where w henceforth denotes the group size. Note that $w = \Delta_1(\mathbf{Q})$ in GS-S, whereas w in GS is a tunable parameter.

We next focus on how GS modifies $\mathcal{G}$ in order to determine the w that leads to the best grouping strategy $\mathbf{g}$. It initially derives a grouping strategy $\mathbf{g}_w$ from $\mathbf{t}_s$ for every group size $w = \{1, 2, \ldots, d\}$. Once again, the size of the last group is $w+(d \mod w)$. Then, GS computes an *estimate* $\tilde{\mathbf{c}}$ of $\mathbf{c}$ using $\mathbf{t}_s$. Next, it *simulates* $\mathcal{M}_\mathbf{g}$ on $\mathbf{g}_w$ and $\tilde{\mathbf{c}}$, in order to derive an *estimate* $\tilde{\mathbf{t}}_w$ of $\mathbf{t}$. Finally, it selects the $\mathbf{g}_w$ that leads to

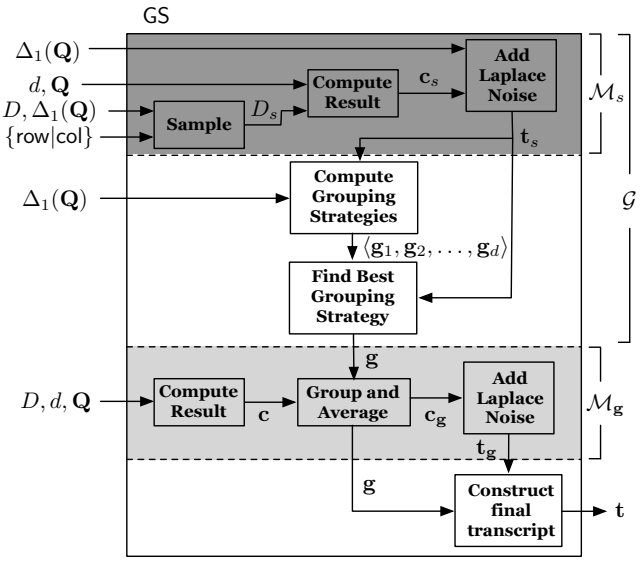


Figure 5: Outline of GS

the *smallest* L_1 norm (absolute error) between $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{t}}_w$ as the optimal strategy $\mathbf{g}$. The above procedure allows $\mathcal{G}$ to select the best $\mathbf{g}_w$, without "looking into" the unperturbed data.

In more detail, $\tilde{\mathbf{t}}_w$ is estimated as follows. $\mathcal{G}$ first calculates $\tilde{\mathbf{c}} = \Delta_1(\mathbf{Q}) \cdot \mathbf{t}_s$. Next, it groups and averages $\tilde{\mathbf{c}}$ using $\mathbf{g}_w$, yielding $\tilde{\mathbf{c}}_{\mathbf{g}_w}$. Then, it derives $\tilde{\mathbf{t}}_{\mathbf{g}_w}$ by adding noise $\langle Lap(\frac{2\cdot\Delta_1(\mathbf{Q})}{\epsilon\cdot w})\rangle^{d_{\mathbf{g}_w}}$ to $\tilde{\mathbf{c}}_{\mathbf{g}_w}$. Finally, it combines $\tilde{\mathbf{t}}_{\mathbf{g}_w}$ with $\mathbf{g}_w$ to get $\tilde{\mathbf{t}}_w$.

We next establish the privacy of GS, after proving the following necessary lemma.

LEMMA 2. *In* GS*, it holds that* $\Delta_1(\mathbf{Q}_\mathbf{g}) \leq \frac{\Delta_1(\mathbf{Q})}{w}$.

PROOF. Similar to Lemma 1, we define u to be a user in D and $m_{u,j}$ the number of 1's that u has in the columns belonging to the j^{th} group of $\mathbf{g}$. The number of groups in $\mathbf{g}$ is $d_\mathbf{g} = \lfloor d/w \rfloor$, and u has at most $\Delta_1(\mathbf{Q})$ 1's in total. Again, it holds that $\sum_{j=1}^{d_\mathbf{g}} m_{u,j} \leq \Delta_1(\mathbf{Q})$. Suppose that we remove u from D to derive neighboring database D'. Observe that now u affects element j of $\mathbf{Q}_\mathbf{g}(D)$ by *at most* $m_{u,j}/w$, since each group consists of *at least* w columns. Using Definition 3, it follows that

$$\Delta_1(\mathbf{Q}_\mathbf{g}) \leq \max_{D\in\mathcal{D}\wedge u\in D} \sum_{j=1}^{d_\mathbf{g}} \frac{m_{u,j}}{w} = \frac{\Delta_1(\mathbf{Q})}{w}$$

which concludes our proof. □

THEOREM 6. GS *satisfies ϵ-differential privacy.*

PROOF. Similar to Theorem 5, Formulae 7 and 9 hold. Moreover, $\mathcal{M}_s$ in GS is identical to that in GS-S with row or column sampling. Therefore, it achieves $\frac{\epsilon}{2}$-differential privacy. Observe that it suffices to prove that $\mathcal{M}_\mathbf{g}$ also achieves $\frac{\epsilon}{2}$-differential privacy, in order to prove that GS guarantees ϵ-differential privacy. Let w be the group size in the determined strategy $\mathbf{g}$. According to Lemma 2, $\Delta_1(\mathbf{Q}_\mathbf{g}) \leq \frac{\Delta_1(\mathbf{Q})}{w}$. Moreover, $\mathcal{M}_\mathbf{g}$ injects noise $Lap(\frac{2\cdot\Delta_1(\mathbf{Q})}{\epsilon\cdot w})$. Consequently, due to Theorem 1, $\mathcal{M}_\mathbf{g}$ achieves $\frac{\epsilon}{2}$-differential privacy, which completes our proof. □

Input: $D, d, \mathbf{Q}, \Delta_1(\mathbf{Q}), k$, sampling_method
Output: $\mathbf{t}$

/* *Module* $\mathcal{G}$ */
1. Get D_s based on sampling_method
2. $\mathbf{c}_s = \mathbf{Q}(D_s)$ and $\mathbf{t}_s = \mathbf{c}_s + \langle Lap(2/\epsilon)\rangle^d$
3. Initialize $I = \langle 1, 2, \ldots, d\rangle$ and sort it based on $\mathbf{t}_s$
4. $\tilde{\mathbf{c}} = \Delta_1(\mathbf{Q}) \cdot \mathbf{t}_s$
5. For $w = 1$ to d
6. Set $d_{\mathbf{g}_w} = \lfloor d/w \rfloor$ and initialize empty $\mathbf{g}_w$ of size $d_{\mathbf{g}_w}$
7. Compute $\mathbf{g}_w$ from sorted I
8. Compute $\tilde{\mathbf{c}}_{\mathbf{g}_w}$ using $\mathbf{g}_w$ and $\tilde{\mathbf{c}}$
9. $\tilde{\mathbf{t}}_{\mathbf{g}_w} = \tilde{\mathbf{c}}_{\mathbf{g}_w} + \langle Lap(\frac{2\cdot\Delta_1(\mathbf{Q})}{\epsilon\cdot w})\rangle^{d_{\mathbf{g}_w}}$
10. Compute $\tilde{\mathbf{t}}_w$ using $\mathbf{g}_w$ and $\tilde{\mathbf{t}}_{\mathbf{g}_w}$
11. Set $w^* = \arg\min_w L_1(\tilde{\mathbf{c}}, \tilde{\mathbf{t}}_w)$ and $\mathbf{g} = \mathbf{g}_{w^*}$

/* *Module* $\mathcal{M}_g$ */
12. Initialize $\mathbf{c} = \mathbf{Q}(D)$ and empty $\mathbf{c}_\mathbf{g}$ of size $d_\mathbf{g}$
13. Compute $\mathbf{c}_\mathbf{g}$ using $\mathbf{g}$ and $\mathbf{c}$
14. Compute $\mathbf{t}_\mathbf{g} = \mathbf{c}_\mathbf{g} + \langle Lap(\frac{2\cdot\Delta_1(\mathbf{Q})}{\epsilon\cdot w})\rangle^{d_\mathbf{g}}$
15. Initialize empty vector $\mathbf{t}$ of size d
16. Compute $\mathbf{t}$ using $\mathbf{g}$ and $\mathbf{t}_\mathbf{g}$

Figure 6: Pseudocode of GS

Our experiments (Section 5) show that our fine-tuning technique effectively determines a close-to-optimal w. More importantly, they indicate that different datasets have different optimal group sizes, which may greatly deviate from $\Delta_1(\mathbf{Q})$. This justifies the importance of the fine-tuning method as compared to the ad hoc selection of GS-R and GS-S to set $w = \Delta_1(\mathbf{Q})$.

4. UTILITY ANALYSIS

In this section we analyze the utility of GS. We also (i) show that column sampling is superior to row sampling, (ii) explain the effect of w (group size), and (iii) qualitatively compare GS with LPA and FPA.

Similar to the error analysis of FPA (see Appendix A), the expected error of the i^{th} published count $\mathbf{t}[i]$ in GS is

$$\begin{aligned} error^i_{\mathsf{GS}} &\leq RE^i_{\mathsf{GS}}(w) + \sqrt{Var(\mathbf{t}[i])} \\ &= RE^i_{\mathsf{GS}}(w) + \sqrt{2}\cdot\frac{2\cdot\Delta_1(\mathbf{Q})}{\epsilon\cdot w} \end{aligned} \quad (13)$$

where $RE^i_{\mathsf{GS}}(w)$ is the error attributed to smoothing, and $\sqrt{Var(\mathbf{t}[i])}$ is the error due to Laplace noise. The smoothing error is affected by the sampling method and the grouping strategy (and, thus, w). It is also *data-dependent* and cannot be rigorously analyzed without making strong (typically unrealistic) assumptions about the data distribution. However, we can still elaborate on why column sampling leads to higher utility than row sampling.

Consider the *total* error of GS for *any* fixed w, i.e.,

$$error_{\mathsf{GS}} = \sum_{i=1}^{d} error^i_{\mathsf{GS}} = \sum_{i=1}^{d} RE^i_{\mathsf{GS}}(w) + d\cdot\sqrt{2}\cdot\frac{2\cdot\Delta_1(\mathbf{Q})}{\epsilon\cdot w}$$

Value $\sum_{i=1}^{d} RE^i_{\mathsf{GS}}(w)$ is the total smoothing error. Its *optimal* value is yielded from the *optimal grouping*, which depends on $\mathbf{c}$. However, GS derives the grouping strategy based on $\mathbf{t}_s$. Note that $\mathbf{t}_s[i]$ is essentially a *biased* estimator of $\mathbf{c}[i]$, which relies on the sampling method. Intuitively, the better the estimation of $\mathbf{t}_s[i]$ for *all* i, the "closer" the grouping strategy computed in GS is to the optimal (the full-fledged proof of this intuition is out of the scope of this work). We next prove that $\mathbf{t}_s[i]$ in column sampling is a better estimator of $\mathbf{c}[i]$ than that in row sampling for *every* i, concluding that column sampling is superior to its row counterpart.

The quality of a biased estimator is typically calculated via its *mean squared error* (MSE) w.r.t. the estimated parameter, that is

$$MSE(\mathbf{t}_s[i], \mathbf{c}[i]) = Var(\mathbf{t}_s[i]) + (Bias(\mathbf{t}_s[i], \mathbf{c}[i]))^2 \quad (14)$$

We first focus on row sampling. $Var(\mathbf{t}_s[i])$ depends on two independent events; sampling and Laplace noise injection. Therefore, it is simply the sum of the variance of sampling, and that of Laplace noise. The Laplace variance is equal to $2\cdot\lambda^2 = 8/\epsilon^2$. Since each row is independently sampled with rate β, each of the $\mathbf{c}[i]$ 1's of column i will be incorporated in $\mathbf{t}_s[i]$ with probability β. Hence, the sampling variance of $\mathbf{t}_s[i]$ is $\mathbf{c}[i]\cdot\beta\cdot(1-\beta)$. Moreover, the expected value of $\mathbf{t}_s[i]$ is $\mathbb{E}(\mathbf{t}_s[i]) = \mathbf{c}[i]\cdot\beta$. The bias of $\mathbf{t}_s[i]$ is equal to $\mathbb{E}(\mathbf{t}_s[i]) - \mathbf{c}[i] = \mathbf{c}[i]\cdot(\beta-1)$. Combining the above, we derive

$$\begin{aligned} MSE_{\mathsf{row}}(\mathbf{t}_s[i], \mathbf{c}[i]) &= \mathbf{c}[i]\cdot\beta\cdot(1-\beta) \\ &\quad + (\mathbf{c}[i]\cdot(\beta-1))^2 + 8/\epsilon^2 \end{aligned} \quad (15)$$

We next turn to column sampling. Similar to the case of row sampling, $Var(\mathbf{t}_s[i])$ incorporates sampling and Laplace variance. The Laplace variance is the same as in row sampling, but the sampling variance is different. The reason is that column sampling uses a different probability for choosing each 1 of column i. Let $j \in [1, \mathbf{c}[i]]$ and p_j be the probability of choosing the j^{th} 1 of column i. The sampling variance is $\sum_{j=1}^{\mathbf{c}[i]} p_j\cdot(1-p_j) = \sum_{j=1}^{\mathbf{c}[i]} p_j - \sum_{j=1}^{\mathbf{c}[i]} p_j^2$. From a straightforward application of the Cauchy-Schwarz inequality, it holds that $\sum_{j=1}^{\mathbf{c}[i]} p_j^2 \geq (\sum_{j=1}^{\mathbf{c}[i]} p_j)^2/\mathbf{c}[i]$. Let $\gamma = (\sum_{j=1}^{\mathbf{c}[i]} p_j)/\mathbf{c}[i]$. Then, the sampling variance becomes smaller than or equal to $\mathbf{c}[i]\cdot\gamma\cdot(1-\gamma)$. The expected value of $\mathbf{t}_s[i]$ is $\sum_{j=1}^{\mathbf{c}[i]} p_j = \gamma\cdot\mathbf{c}[i]$. The bias then becomes $\mathbf{c}[i]\cdot(\gamma-1)$. Combining all the above, we get

$$MSE_{\mathsf{col}}(\mathbf{t}_s[i], \mathbf{c}[i]) \leq \mathbf{c}[i]\cdot\gamma\cdot(1-\gamma) \quad (16)$$

$$+ (\mathbf{c}[i]\cdot(\gamma-1))^2 + 8/\epsilon^2 \quad (17)$$

Observe that the right-hand sides of Formulae 15 and 16 can be expressed as a function $f(x)$, where $x = \beta$ in row sampling and $x = \gamma$ in column sampling. Moreover, it can be shown that $f(x)$ is *monotonically decreasing* when $\mathbf{c}[i]$ is a positive integer (we omit the tedious calculations due to space limitation). Therefore, in order to prove that MSE is lower in column than in row sampling, it suffices to show that $\gamma \geq \beta$.

We first calculate that

$$\begin{aligned} \beta = \frac{e^{\epsilon/2}-1}{e^{\epsilon\cdot\Delta_1(\mathbf{Q})/2}-1} &\leq \frac{e^{\epsilon/2}}{e^{\epsilon\cdot\Delta_1(\mathbf{Q})/2}} \\ &\leq \frac{\epsilon/2}{\epsilon\cdot\Delta_1(\mathbf{Q})/2} = \frac{1}{\Delta_1(\mathbf{Q})} \end{aligned} \quad (18)$$

In column sampling, p_j is *at least* $1/\Delta_1(\mathbf{Q})$, because a row has *at most* $\Delta_1(\mathbf{Q})$ 1's and we randomly choose one of them. Therefore, it holds that

$$\gamma = \frac{\sum_{j=1}^{\mathbf{c}[i]} p_j}{\mathbf{c}[i]} \geq \frac{\mathbf{c}[i]\cdot\frac{1}{\Delta_1(\mathbf{Q})}}{\mathbf{c}[i]} = \frac{1}{\Delta_1(\mathbf{Q})} \quad (19)$$

From Formulae 18 and 19 we derive that $\gamma \geq \beta$, concluding that $MSE_{\mathsf{col}} \leq MSE_{\mathsf{row}}$.

It is now apparent that the effect of w is *data-dependent*. In particular, Formula 13 suggests that as w increases, it diminishes the effect of $\Delta_1(\mathbf{Q})/\epsilon$ and the error due to Laplace noise decreases. Nevertheless, the larger the w, the stronger the smoothing error, which greatly depends on the distribution of the counts in $\mathbf{c}$. The above justifies our fine-tuning mechanism in Section 3, which operates on the *given* dataset without compromising privacy.

Compared to LPA, both GS and FPA introduce less Laplace noise (tunable by w and ρ, respectively), but their total error also depends on the reconstruction error (due to smoothing and DFT approximation, respectively). The latter is *data-dependent*, which essentially renders the mechanisms incomparable, *unless* the dataset under sanitization is thoroughly studied. FPA works great for datasets where DFT approximation is effective, but it is expected to perform poorly in case the counts in $\mathbf{c}$ have no natural ordering. On the contrary, GS groups similar counts and minimizes the perturbation error, *irrespectively* of the ordering in $\mathbf{c}$. Therefore, we expect GS to generally outperform both FPA and LPA. Our experiments in the next section confirm our claims.

5. EXPERIMENTAL EVALUATION

We first explain our experimental setup. Subsequently, we provide a comparison between column and row sampling in GS-S. Next, we assess the fine-tuning technique of GS. Finally, we compare our schemes in terms of utility against our competitors, namely LPA and FPA.

Setup We experimented with four real datasets, henceforth referred to as *Checkin*, *Document*, *Netflix*, and *Trajectory*. These datasets feature considerably different characteristics, which are outlined in Table 2. *Checkin* contains user "checkins" from the Gowalla geo-social network[1]. We modeled each user as a row, and each column as a location. A cell value is non-zero if the user visited the respective location at least once. *Document* is the PubMed[2] dataset. Rows stand for documents/abstracts on life sciences and biomedical topics, columns represent vocabulary terms, and a non-zero cell stores the frequency of the term in the respective document. *Netflix* is the training dataset used in the Netflix Prize competition[3]. Rows are users, columns are movies, and a non-zero cell contains the rating of the movie given by the corresponding user. *Trajectory* consists of GPS traces gathered by the GeoPKDD project[4], mapped on a road network. Rows correspond to trajectories, columns are road network nodes, and a non-zero cell represents a transition from the respective node.

For all datasets, the query under consideration is $\mathbf{Q}$ as defined in our model in Section 2.1, i.e., the vector to be privately published contains all the column counts (that is, number of non-zero cells). We assume that $\Delta_1(\mathbf{Q})$ is the maximum number of non-zero values observed in a row. Moreover, we use the *optimal* ρ for FPA without sacrificing privacy budget, noting that this renders it non-private. In this sense, we *overestimate* the performance of FPA.

[1] http://snap.stanford.edu/data/loc-gowalla.html

[2] http://archive.ics.uci.edu/ml/datasets/Bag+of+Words

[3] http://www.netflixprize.com/

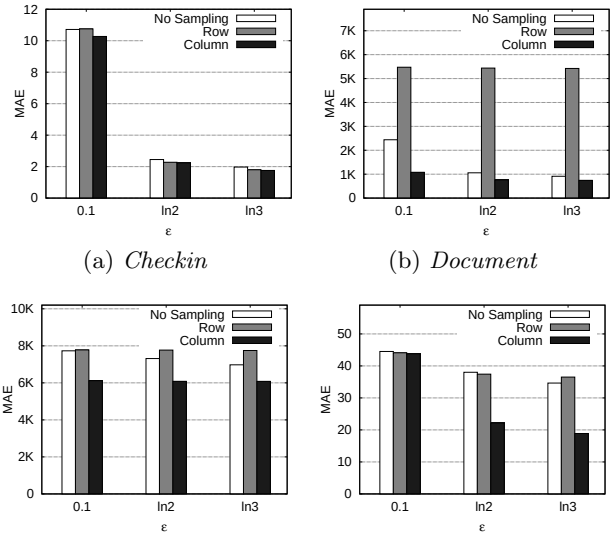
[4] http://www.geopkdd.eu

Table 2: Dataset characteristics

Dataset	Rows	Columns	$\Delta_1(\mathbf{Q})$	Avg. count
Checkin	196,591	5,977,758	2,175	1.08
Document	8,200,000	141,043	436	5,175.72
Netflix	480,189	17,770	17,000	5,654.50
Trajectory	62,956	14,021	55	43.54

We implemented all mechanisms in Java. We conducted the experiments on a machine equipped with a 2.53GHz Intel Core i5 CPU and 4GB RAM, running Windows 7. We ran each experiment 200 times, and reported the average error. The error metrics we used are the *Mean Absolute Error* (MAE) and the *Mean Relative Error* (MRE) with sanity bound equal to 0.1% of the database size [25].

Column vs. row sampling in GS-S Figure 7 evaluates the MAE of row and column sampling in GS-S, as well as the case where no sampling was used at all, varying parameter ϵ for the four datasets. In all datasets, column sampling reduces the Laplace noise injected to $\mathbf{c}_s$. At the same time, the resulting sample captures substantial information, yielding a very representative $\mathbf{c}_s$. On the other hand, although row sampling adds the same noise as column sampling, its sample is much poorer. Consequently, column sampling outperforms both row and no sampling at all.



(a) *Checkin* (b) *Document*

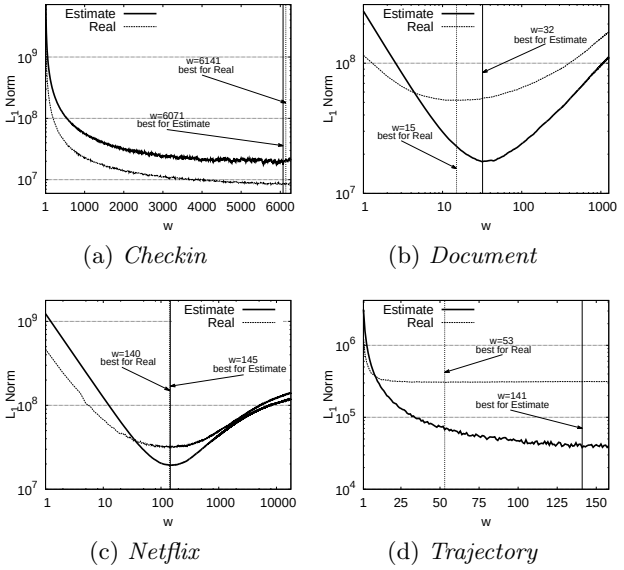
(c) *Netflix* (d) *Trajectory*

Figure 7: Comparison of sampling methods

Interestingly, row sampling does not seem to have significant gains over no sampling at all. It is noteworthy to describe the considerably poor performance of row sampling in Figure 7(b) for *Document*. When no sampling is used, $\mathbf{c}_s$ is equal to $\mathbf{c}$. Moreover, the noise added due to sensitivity $\Delta_1(\mathbf{Q}) = 436$ does not significantly alter the already large counts of $\mathbf{c}_s$ (note that the average count in *Document* is $5,175.72$). As such, the yielded $\mathbf{t}_s$ leads to a quite effective $\mathbf{g}$. In contrast, due to the small sample size in row sampling, the quality of $\mathbf{c}_s$ drastically decreases. The resulting ineffective $\mathbf{g}$ produces very strong smoothing perturbation especially in *Document*, because the counts feature large variance.

In overall, column sampling is the best sampling method in all scenarios. Its savings reach up to $\sim 56\%$ in *Document* in Figure 7(b) for $\epsilon = 0.1$. Henceforth, we implement GS-S and GS with column sampling.

Fine-tuning GS The choice of w by GS is based on the L_1 norm of estimators $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{t}}_\mathbf{w}$ of $\mathbf{c}$ and $\mathbf{t}$, respectively (Line 11 in Figure 6). Figure 8 assesses for all datasets and varying w the accuracy of this *estimated* L_1 norm versus the *real* L_1 norm between $\mathbf{c}$ and $\mathbf{t}$, in order to determine the quality of the selection of w. We set $\epsilon = \ln 2$. What is of importance in this experiment is the similarity between the real L_1 error given by the estimated best w, and that yielded by the real best w (i.e., the similarity between the values where the two vertical lines intersect with the 'Real' curve). In other words, we wish to test whether the w chosen by the fine-tuning module of GS leads to similar performance to the case GS uses the actual optimal w. In all scenarios, the relative difference between the two L_1 errors ranges between 1%-3.4%. This suggests that our fine-tuning technique is quite accurate. Moreover, observe that the best w greatly depends on the dataset.



(a) *Checkin* (b) *Document*

(c) *Netflix* (d) *Trajectory*

Figure 8: L_1 norm vs. group size w

Comparison of mechanisms Figures 9 and 10 display the MAE and MRE, respectively, of all mechanisms for all datasets and for $\epsilon \in \{0.1, \ln 2, \ln 3\}$. GS outperforms LPA and FPA in *all* scenarios. The gains against LPA are most prominent in the *Checkin* dataset, where GS features up to more than 3 orders of magnitude smaller MAE and MRE than LPA. On the other hand, the largest advantage of GS over FPA is observed in *Document* and *Netflix*, where GS exhibits up to one order of magnitude better MAE and MRE than FPA. In fact, FPA behaves like our GS-R (random grouping) mechanism, because DFT is quite ineffective in these datasets.

In addition, GS is always superior to GS-R. Observe that GS-R becomes even worse than LPA in *Document*, due to the strong smoothing perturbation stemming from the poor grouping. Furthermore, GS is superior to GS-S mainly in *Netflix*, where it achieves reduction up to a factor of 5 in MAE and MRE when $\epsilon = \ln 3$. However, the two mechanisms are comparable in *Trajectory*. The reason is that the tuning of w does not induce a substantial reduction in MAE and MRE compared to the fixed $w = \Delta_1(\mathbf{Q})$ of GS-S. This is actually apparent in Figure 8 for the L_1 norm.

In overall, GS inflicts relative error of 0.7%, 2.4%, 73%, and 27% for *Checkin*, *Document*, *Netflix*, and *Trajectory*, respectively, when $\epsilon = \ln 2$. Observe that even in the case of *Netflix*, where the sensitivity is extreme, GS still performs decently considering the 3,187% and 682% error of LPA and FPA, respectively. Finally, our largest gains against LPA are observed for smaller ϵ values, whereas against FPA for larger ϵ values. To conclude GS is an ideal choice for any dataset and privacy setting.

6. CONCLUSION

In this paper we introduced GS, a novel method that offers ϵ-differential privacy in count query result publication. Our mechanism groups the counts and smooths them via averaging, prior to injecting Laplace noise. This diminishes the sensitivity and, thus, drastically decreases the Laplace noise scale. The grouping technique relies on a novel sampling mechanism, which produces a grouping strategy that minimizes the smoothing perturbation. We provided rigorous proofs, optimizations, and a detailed experimental evaluation on four real datasets. Our results showed that GS outperforms previous work, and proved its practicality in a wide set of applications.

7. REFERENCES

[1] Foursquare. `https://foursquare.com/`.

[2] G. Cormode, C. M. Procopiuc, D. Srivastava, and T. T. L. Tran. Differentially private publication of sparse data. In *ICDT*, 2012.

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

[4] C. Dwork. A firm foundation for private data analysis. *CACM*, 54(1):86–95, 2011.

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

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

[7] M. Götz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke. Publishing search logs: A comparative study of privacy guarantees. *TKDE*, 24(3):520–532, 2012.

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

[9] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. In *PVLDB*, 2010.

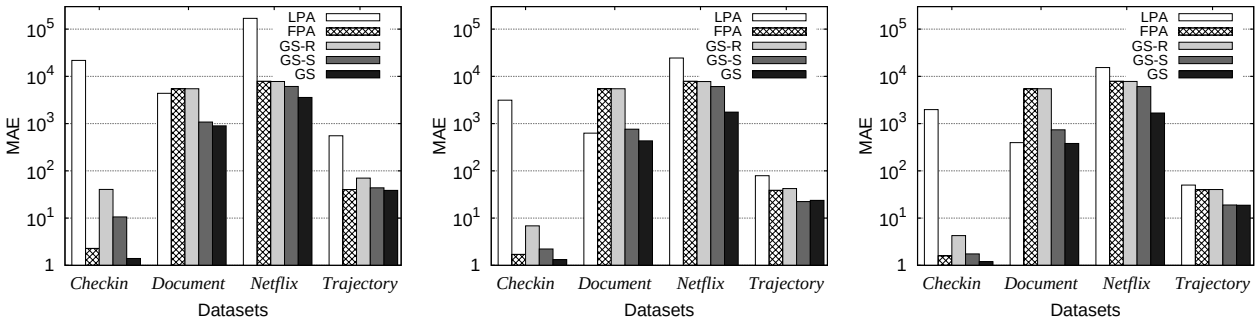
[10] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *PODS*, 2010.

[11] N. Li, W. Qardaji, and D. Su. On sampling, anonymization, and differential privacy: Or, k-anonymization meets differential privacy. *Arxiv preprint arXiv:1101.2604*, 2011.

[12] Y. D. Li, Z. Zhang, M. Winslett, and Y. Yang. Compressive mechanism: Utilizing sparse representation in differential privacy. In *WPES*, 2011.

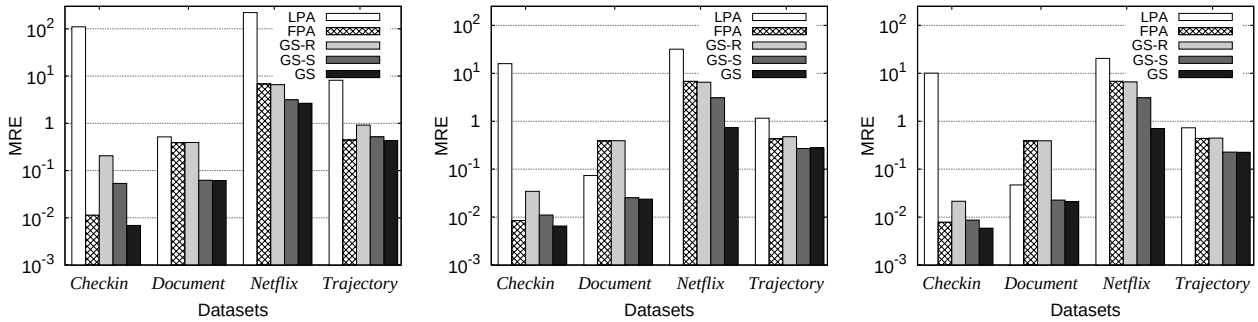
[13] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *ICDE*, 2008.

[14] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *TKDD*, 1(1), 2007.



(a) $\epsilon = 0.1$ (b) $\epsilon = \ln 2$ (c) $\epsilon = \ln 3$

Figure 9: MAE vs. datasets



(a) $\epsilon = 0.1$ (b) $\epsilon = \ln 2$ (c) $\epsilon = \ln 3$

Figure 10: MRE vs. datasets

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

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

[17] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. Technical Report MSR-TR-2009-186, Microsoft Research, 2009.

[18] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *SIGMOD*, 2010.

[19] A. Roth and T. Roughgarden. Interactive privacy via the median mechanism. In *STOC*, 2010.

[20] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information. In *PODS*, 1998.

[21] R. Sarathy and K. Muralidhar. Evaluating Laplace noise addition to satisfy differential privacy for numeric data. *TDP*, 4(1):1–17, 2011.

[22] K. Wang and B. C. M. Fung. Anonymizing sequential releases. In *KDD*, 2006.

[23] X. Xiao, G. Bender, M. Hay, and J. Gehrke. iReduct: Differential privacy with reduced relative errors. In *SIGMOD*, 2011.

[24] X. Xiao and Y. Tao. Personalized privacy preservation. In *SIGMOD*, 2006.

[25] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. *TKDE*, 23(8):1200–1214, 2011.

[26] Y. Xiao, L. Xiong, and C. Yuan. Differentially private data release through multidimensional partitioning. In *SDM*, 2010.

[27] J. Xu, Z. Zhang, X. Xiao, Y. Yang, and Y. Ge. Differentially private histogram publication. In *ICDE*, 2012.

[28] G. Yuan, Z. Zhang, M. Winslett, X. Xiao, Y. Yang, and Z. Hao. LowRank mechanism: Optimizing batch queries under differential privacy. *PVLDB*, 2012.

APPENDIX

A. PROOF OF THEOREM 2

Let $\mathbf{F}$ denote the DFT coefficients of vector $\mathbf{Q}(D)$, and $\mathbf{F}^\rho$ the first ρ coefficients of $\mathbf{F}$. For the sake of simplicity, we abuse notation and write $\Delta_p(\mathbf{F})$ and $\Delta_p(\mathbf{F}^\rho)$ to denote the L_p sensitivity of a query that outputs $\mathbf{F}$ and $\mathbf{F}^\rho$, respectively. Then, $\Delta_1(\mathbf{F}^\rho)$ is essentially the L_1 sensitivity of our query in FPA. From Parseval's Theorem in the discrete domain, it holds that $\frac{1}{d} \cdot L_2(\mathbf{F})^2 = L_2(\mathbf{Q}(D))^2$, which is equivalent to $\Delta_2(\mathbf{F}) = \sqrt{d} \cdot \Delta_2(\mathbf{Q})$. Since $\rho \leq d$, $\Delta_2(\mathbf{F}^\rho) \leq \Delta_2(\mathbf{F})$. Moreover, due to a standard inequality between L_1 and L_2 norms [18], $\Delta_1(\mathbf{F}^\rho) \leq \sqrt{\rho} \cdot \Delta_2(\mathbf{F}^\rho)$. In addition, $\Delta_2(\mathbf{Q}) = \sqrt{\Delta_1(\mathbf{Q})}$ in our setting; for every i it holds that $|\mathbf{Q}(D)_i - \mathbf{Q}(D')_i| = (\mathbf{Q}(D)_i - \mathbf{Q}(D')_i)^2$, because $\mathbf{Q}(D)_i$ and $\mathbf{Q}(D')_i$ are either equal or differ by *exactly* 1. Combining all the above, we derive that $\Delta_1(\mathbf{F}^\rho) \leq \sqrt{\rho \cdot d \cdot \Delta_1(\mathbf{Q})}$. Consequently, from Theorem 1, FPA satisfies ϵ-differential privacy when $\lambda = \frac{\sqrt{\rho \cdot d \cdot \Delta_1(\mathbf{Q})}}{\epsilon}$.

Let $\mathbf{t}$ be the published transcript in FPA. To calculate the error for each $\mathbf{t}[i]$, we follow an identical procedure to [17] (the long version of [18]), and derive that $error^i_{\mathsf{FPA}} \leq RE^i_{\mathsf{FPA}}(\rho) + \sqrt{Var(\mathbf{t}[i])}$. Next, we calculate $\sqrt{Var(\mathbf{t}[i])}$ similarly to [17] using the λ we computed above, i.e, $Var(\mathbf{t}[i]) = \frac{\rho \cdot 2 \cdot (\lambda)^2}{d^2} = 2 \cdot \frac{\rho^2 \cdot \Delta_1(\mathbf{Q})}{\epsilon^2 \cdot d}$. Thus, the error becomes $error^i_{\mathsf{FPA}} \leq RE^i_{\mathsf{FPA}}(\rho) + \sqrt{2} \cdot \frac{\rho \cdot \sqrt{\Delta_1(\mathbf{Q})}}{\epsilon \cdot \sqrt{d}}$, which concludes our proof.