Differentially Private Histogram Publishing through Lossy Compression

Gergely Acs
INRIA
France
gergely.acs@inria.fr

Claude Castelluccia
INRIA
France
claude.castelluccia@inria.fr

Rui Chen*
Concordia University
Montreal, QC, Canada
ru_che@encs.concordia.ca

***Abstract*—Differential privacy has emerged as one of the most promising privacy models for private data release. It can be used to release different types of data, and, in particular, histograms, which provide useful summaries of a dataset. Several differentially private histogram releasing schemes have been proposed recently. However, most of them directly add noise to the histogram counts, resulting in undesirable accuracy.**

In this paper, we propose two sanitization techniques that exploit the inherent redundancy of real-life datasets in order to boost the accuracy of histograms. They lossily compress the data and sanitize the compressed data. Our first scheme is an optimization of the Fourier Perturbation Algorithm (FPA) presented in [13]. It improves the accuracy of the initial FPA by a factor of 10. The other scheme relies on clustering and exploits the redundancy between bins. Our extensive experimental evaluation over various real-life and synthetic datasets demonstrates that our techniques preserve very accurate distributions and considerably improve the accuracy of range queries over attributed histograms.

***Keywords*-Differential privacy, histogram, lossy compression, Fourier transform, clustering**

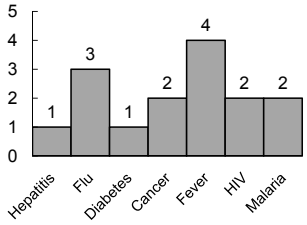
I. Introduction

With the general trend of digitalization, data from various application domains, such as retailing companies, healthcare departments, public transit agencies and online social networks, have been widely generated and collected in recent years, and have opened the possibility to conduct different data mining tasks. When aggregated, such data can help understand complex processes (e.g., the spread of viruses), build better transportation systems, and prevent traffic congestion, among others. While the benefits provided by these data are indisputable, they unfortunately pose a considerable threat to privacy. This not only negatively impacts people's daily lives, but also hinders the advance of research. Privacy is so important that companies and researchers are reluctant to publish datasets by fear of being held responsible for potential privacy breaches. As a result, only very few datasets are released and available. This limits our ability to analyze such data to derive information that could benefit the general public. It is then urgent to develop tools for privately releasing datasets.

This paper considers the problem of private histogram releasing. A histograms is typically defined over a specific domain and a dataset. It summarizes the occurrence counts of domain values over the dataset. For example, if the domain is a set of diseases D (such as cancer, flu, HIV, hepatitis, etc.), then a histogram over a patient dataset assigns the number of patients with disease $d \in D$ in the dataset to d. Given the disease information of 15 patients in Figure 1 (a), Figure 1 (b) presents its corresponding histogram on the attribute $Disease$. This histogram provides useful statistical summaries of the disease distribution in a given population. However, a histogram inevitably leaks sensitive information about the underlying dataset. For example, if the adversary knows the diseases of 14 patients, he can easily infer the disease of the last patient from the released histogram.

Name	Disease
Bob	Flu
Ann	HIV
Frank	Flu
Steven	Flu
Lucy	Cancer
David	Fever
Alice	Fever
...	...

(a) Patient table



(b) The histogram

Fig. 1. Sample patient table and its corresponding histogram

To prevent such information leakage, histograms must be sanitized before release. Although different privacy models can be used, one of the only models that provide provable privacy guarantees is *differential privacy* [3]. The main idea of differential privacy is to add sufficient noise to each histogram count so that an adversary cannot decide whether the information of a particular record owner was used to generate the sanitized histogram or not. The variance of the noise injected must be carefully calibrated to both the *sensitivity* of a function (i.e., the maximal change of the function due to the inclusion/exclusion of a single record in a dataset) and a desired privacy level ε. For a more stringent privacy requirement (e.g., $\varepsilon = 0.01$ or $\varepsilon = 0.1$), the added noise often substantially degrades the accuracy of the sanitized histogram.

In this paper, instead of directly adding noise to histogram counts, we consider *lossy compression* techniques that exploit the lower sensitivity of compressed data to reduce the magnitude of noise. A lossy compression mechanism first lossily compresses the data, then adds noise calibrated to the (lower) sensitivity of the compressed data. In this process,

* This work was done when the author was on an internship at INRIA

there are two sources of errors: the error introduced by lossy compression, called *reconstruction error*, and the error due to added noise, called *perturbation error*. Hence, the final utility of a released histogram is determined by the trade-off between reconstruction error and perturbation error. A more lossy compression increases reconstruction error, but requires less noise to be injected due to the decreased sensitivity; a less lossy compression affects the two types of errors in the opposite way. Since in practice many real-life data are highly-compressible (e.g., contain many similar counts), the idea of lossy compression exhibits great promise for achieving better utility.

Following this idea, we propose two novel algorithms that significantly outperform the state-of-the-art techniques [3], [13], [14], [7], [5], [16]. We summarize our contributions as follows.

1) We propose an enhanced version of the Fourier Perturbation Algorithm (FPA) defined in [13], called *EFPA*. Our scheme applies the Fourier transform to a histogram and compresses it by removing high-frequency components using the exponential mechanism. We improve the performance of FPA by designing a more accurate score function for the exponential mechanism and exploiting the intrinsic correlation among the Fourier coefficients of real-valued histograms. Experimental results show a utility improvement by a factor of 10.
2) We propose *P-HPartition* that uses a divisible hierarchical clustering (partitioning) scheme to compress histograms. The intuition is that histogram bins belonging to the same cluster have similar counts, and hence can be approximated by their mean value (i.e., cluster center). It is often sufficient to release only the noisy cluster centers, which have a smaller sensitivity. *P-HPartition* outperforms the existing schemes over different datasets in terms of both KL-divergence and range count queries.

The rest of the paper is structured as follows: Section II summarizes the related works. Section III presents the preliminaries. Section IV describes our enhanced Fourier Perturbation Algorithm (EFPA). Section V motivates the use of clustering for accuracy boosting and presents our clustering-based sanitization algorithm, *P-HPartition*. Section VI reports our experimental results. Section VII provides a discussion on lossy compression. Section VIII concludes the paper.

II. Related Work

Due to the importance of histograms, there have been several recent works [3], [1], [14], [7], [9], [15], [13], [16], [10] studying histogram release under differential privacy. Dwork et al. [3] propose the Laplace mechanism, which lays the ground for all subsequent works. As a naive solution, a histogram can be differentially privately released by adding independent Laplace noise to each bin. However, the resulting utility of this solution is usually poor. Barak et al. [1] study how to release accurate contingency tables under differential privacy. They apply Laplace mechanism on the Fourier coefficients of an input database and employ linear programming to create non-negative contingency tables. Xiao et al. [14] propose a histogram publishing technique called *Privelet*, which is based on wavelet transforms. Privelet first applies a wavelet transform on the input data and then adds polylogarithmic noise to the transformed data. Hay et al. [7] point out that constrained inference, as a post-processing step, is able to improve the accuracy of histogram queries. Given a set of consistency constraints, they show that for both unattributed and universal histograms, it is possible to enforce these constraints on the noisy values returned by Laplace mechanism in order to boost utility. Li et al. [9] propose the *matrix mechanism*, which generalizes the techniques in [14] and [7]. Given a workload of queries, the matrix mechanism generates a different set of queries (known as a *query strategy*), on which Laplace noise is added. The answers to the workload queries are then calculated based on the noisy answers of the query strategy. This process involves a more complex combination of linear noise, which allows more accurate query answers. Xiao et al. [15] first generate a synthetic histogram by adding Laplace noise to each bin of a histogram, and then conduct a kd-tree based partitioning strategy to identify partitions that are close to the uniform distribution in order to minimize approximation errors. Rastogi and Nath [13] show that, for n queries over time-series, the magnitude of noise can be reduced by just retaining the first k Fourier coefficients of the true query answers. In this case, Laplace noise is only added to k coefficients, resulting in much smaller Laplace noise at the cost of small reconstruction error. Xu et al. [16] exploit the idea of clustering to lower the magnitude of noise in a released histogram, however, as discussed in Section V-C, their approaches suffer from several drawbacks. Compared with the above works [3], [1], [14], [7], [9], [15], [13], [16], the major contribution of our paper is the significant utility improvement, as demonstrated in Section VI. Another recent sanitization technique based on the idea of compression was proposed in [10]. However, it is only applicable to sparse datasets, and in general, cannot exploit other types of redundancy in datasets (e.g., the similarity among non-zero counts).

III. Preliminaries

A. Attributed and Unattributed Histograms

Histograms are commonly used in statistical analysis to provide a summarized representation of the distribution of an attribute in a database. Given an attribute X with the value set $\mathcal{V}$ (either *numerical* or *nominal*) in a database $\mathcal{D}$, one can construct a frequency vector of size $|\mathcal{V}|$ with the i^{th} element being the number of tuples $t \in \mathcal{D}$ with $t.X = v_i \in \mathcal{V}$. A histogram H over the attribute X is constructed by partitioning the frequency vector into a set of *bins* $\{H_1, \cdots, H_n\}$, where each H_i specifies a range of values it covers, and assigns each value a representative count (i.e., the average). The bins are usually specified as consecutive, non-overlapping intervals of the attribute and satisfy the condition: $|\mathcal{D}| = \sum_{i=1}^{n} H_i$, where $|\mathcal{D}|$ is the total number of records in $\mathcal{D}$.

In practice, two types of histograms are widely used, namely *unattributed histogram* and *attributed histogram* [7]. In an

unattributed histogram, the semantic meaning of each bin is irrelevant to the analysis, and, therefore, the histogram can be viewed as a multiset of counts. An unattributed histogram is typically used to study the distribution of a given attribute and to answer different aggregate queries such as the mean or variance. In contrast, an attributed histogram retains the semantic meaning of each bin and can be used to answer various kinds of queries (e.g., range queries). In this paper, we propose different algorithms for both types of histograms.

B. Differential Privacy

Differential privacy, in general, requires that the outcome of any computation be insensitive to the change of a single record. It follows that any information that can be learned from the database with a record can also be learned from the one without this record. Consequently, for a record owner, it means that any privacy breach will not be a result of participating in the database with high probability. Formally, differential privacy [3] is defined below.

Definition 1 (Differential Privacy) *A privacy mechanism $\mathcal{A}$ gives ε-differential privacy if for any database $\mathcal{D}_1$ and $\mathcal{D}_2$ differing on at most one record, and for any possible output $O \in Range(\mathcal{A})$,*

$$Pr[\mathcal{A}(\mathcal{D}_1) = O] \leq e^{\varepsilon} \times Pr[\mathcal{A}(\mathcal{D}_2) = O]$$

where the probability is taken over the randomness of $\mathcal{A}$. ■

Two principal techniques for achieving differential privacy are *Laplace mechanism* [3] (also known as Laplace Perturbation Algorithm (LPA)) and *exponential mechanism* [12]. A fundamental concept of both techniques is the *global sensitivity* of a function [3] that maps underlying databases to (vectors of) reals.

Definition 2 (Global Sensitivity) *For any function $f : \mathcal{D} \rightarrow \mathbb{R}^d$, the sensitivity of f is $\Delta f = \max_{\mathcal{D}_1,\mathcal{D}_2} ||f(\mathcal{D}_1) - f(\mathcal{D}_2)||_1$ for all $\mathcal{D}_1, \mathcal{D}_2$ differing in at most one record.* ■

The global sensitivity is also called as L_1-sensitivity due to the L_1-norm used in its definition and is denoted by $\Delta_1 f$. Similarly, the L_2-sensitivity $\Delta_2 f$ of a function f, which is used later in this paper, is defined by the L_2-norm $||\cdot||_2$.

Laplace mechanism (LPA). For the analysis whose outputs are real, a standard mechanism to achieve differential privacy is to add Laplace noise to the true output of a function. Dwork et al. [3] propose the Laplace mechanism which takes as inputs a database $\mathcal{D}$, a function f, and the privacy parameter ε. The noise is generated according to a Laplace distribution with the probability density function $p(x|\lambda) = \frac{1}{2\lambda} e^{-|x|/\lambda}$, where λ is determined by both Δf and the desired privacy parameter ε.

Theorem 1 *For any function $f : \mathcal{D} \rightarrow \mathbb{R}^d$, the mechanism $\mathcal{A}$*

$$\mathcal{A}(\mathcal{D}) = f(\mathcal{D}) + \langle \mathcal{L}_1(\Delta f/\varepsilon), \ldots, \mathcal{L}_d(\Delta f/\varepsilon) \rangle$$

gives ε-differential privacy, where $\mathcal{L}_i(\Delta f/\varepsilon)$ are i.i.d Laplace variables with scale parameter $\Delta f/\varepsilon$. ■

Exponential mechanism. For the analysis whose outputs are not real or make no sense after adding noise, McSherry and Talwar [12] propose the exponential mechanism that selects an output from the output domain, $r \in \mathcal{R}$, by taking into consideration its score of a given utility function u in a differentially private manner. The exponential mechanism assigns exponentially greater probabilities of being selected to outputs of higher scores so that the final output would be close to the optimum with respect to u. The chosen utility function u should be insensitive to changes of any particular record, that is, has a low sensitivity. Let the sensitivity of u be $\Delta u = \max_{\forall r, \mathcal{D}_1, \mathcal{D}_2} |u(\mathcal{D}_1, r) - u(\mathcal{D}_2, r)|$.

Theorem 2 *Given a utility function $u : (\mathcal{D} \times \mathcal{R}) \rightarrow \mathbb{R}$ for a database $\mathcal{D}$, the mechanism $\mathcal{A}$,*

$$\mathcal{A}(\mathcal{D}, u) = \left\{ \textit{return } r \textit{ with probability} \propto \exp\left(\frac{\varepsilon u(\mathcal{D}, r)}{2\Delta u}\right) \right\}$$

gives ε-differential privacy. ■

The definition of differential privacy enjoys two *composition properties* [11]: *sequential composition* and *parallel composition*, which specify the privacy guarantees in a sequence of computation.

C. Utility Metrics

In this paper, we measure the utility of a released histogram in terms of *mean squared error* (MSE) and *Kullback-Leibler divergence* (KL-divergence).

Mean squared error. The utility of a set of range count queries $\mathbf{Q} = \{Q_1, Q_2, \cdots, Q_n\}$ over a sanitized histogram $\hat{H}$ is commonly measured by the mean squared error (MSE). The MSE of $\mathbf{Q}$ is formalized as $\text{MSE}(H, \hat{H}, \mathbf{Q}) = \frac{\sum_{i=1}^{n} (Q_i(\hat{H}) - Q_i(H))^2}{n}$, where $Q_i(H)$ (or $Q_i(\hat{H})$) returns the answer of Q_i on H (or $\hat{H}$).

KL-divergence. For many applications, it is of importance to make sure that the sanitized histogram $\hat{H}$ has a similar probability distribution as that of H. The difference of the two probability distributions of H and $\hat{H}$ could be measured by KL-divergence: $D_{KL}(H||\hat{H}) = \sum_{i=1}^{n} H_i \log \frac{H_i}{\hat{H}_i}$, where n is the number of bins in H. If $H = \hat{H}$, then $D_{KL}(H||\hat{H}) = 0$. We follow the standard convention that $0 \log 0 = 0$.

IV. Fourier Transform Based Sanitization

Consider a histogram H with bins $\{H_1, H_2, \ldots, H_n\}$. One naive solution to our problem is using LPA to add independent Laplace noise to each bin's count and to release the results: $\hat{H} = \{H_1 + \mathcal{L}(1/\varepsilon), \ldots, H_n + \mathcal{L}(1/\varepsilon)\}$, where the sensitivity of a count query is 1 (because each bin is independent of others). Yet this simple approach usually leads to excessive noise, rendering the released histogram useless.

Rastogi and Nath [13] indicate that Fourier perturbation could be an effective tool for reducing noise in *time-series* under differential privacy. In this paper, we consider a histogram as a time-series and develop an enhanced Fourier Perturbation Algorithm (EFPA), which substantially reduces the magnitude of added noise.

A. Background: Perturbation via Fourier Transform

1) Discrete Fourier Transform: The discrete Fourier transform (DFT) is an invertible, linear transformation that decomposes a function in the time domain into its constituent frequencies. It has been widely used in image processing and signal processing. Given an n-dimensional vector of real or complex numbers $\mathbf{X} = \langle X_0, \ldots, X_{n-1} \rangle$, DFT transforms $\mathbf{X}$ into another n-dimensional vector of complex numbers $\mathbf{F} = \langle F_0, \ldots, F_{n-1} \rangle$ according to the formula given below:

$$F_i = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} X_j \cdot e^{-\frac{2\pi}{n} ji\sqrt{-1}}.$$

$\mathbf{X}$ can be losslessly recovered by applying the inverse discrete Fourier transform (IDFT) from $\mathbf{F}$ as follow:

$$X_i = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} F_j \cdot e^{+\frac{2\pi}{n} ji\sqrt{-1}}.$$

If $\mathbf{X}$ is real-valued (as it is in our case),

$$F_i = F^*_{n-i}, \text{ for } i = \begin{cases} 1, \ldots, (n-1)/2 & \text{if } n \text{ is odd} \\ 1, \ldots, n/2 - 1 & \text{if } n \text{ is even} \end{cases}$$

where F^*_{n-i} denotes the complex conjugate of F_{n-i}. Therefore, the output of DFT is half-redundant, and the vector $\text{DFT}^{\mathsf{real}}(\mathbf{X}) = \langle F_0, \ldots, F_m \rangle$ is sufficient to recover $\mathbf{X} = \text{IDFT}^{\mathsf{real}}(\text{DFT}^{\mathsf{real}}(\mathbf{X}))$, where $m = (n+1)/2$ if n is odd, or $m = n/2 + 1$ otherwise. In practice, $\text{DFT}^{\mathsf{real}}$ and $\text{IDFT}^{\mathsf{real}}$ can be efficiently computed using a fast Fourier transform (FFT) algorithm in $O(n \log n)$.

2) Basic Fourier Perturbation Algorithm (FPA): The basic Fourier Perturbation Algorithm (FPA) [13] works as follows:

1) Compute the DFT coefficients $\mathbf{F} = \text{DFT}(H)$ of a given histogram H with length n by discrete Fourier transform. The length of $\mathbf{F}$ is also n.
2) Remove the last $n-k$ coefficients from $\mathbf{F}$, which correspond to the high-frequency components in H, whereas the first k elements of $\mathbf{F}$, denoted by $\mathbf{F}^k$, preserve the low frequencies in H, and therefore represent the high-level trends of H. Note that k is an input to the algorithm.
3) Generate the noisy version of $\mathbf{F}^k$, denoted by $\hat{\mathbf{F}}^k$, by Laplace mechanism: add i.i.d Laplace noise $\mathcal{L}(\sqrt{k}/\varepsilon)$ to each coefficient in $\mathbf{F}^k$.
4) Pad $\hat{\mathbf{F}}^k$ to be a n-dimensional vector by appending $n-k$ zeros, which is denoted by $\text{PAD}^n(\hat{\mathbf{F}}^k)$. Finally, the inverse DFT is applied to $\text{PAD}^n(\hat{\mathbf{F}}^k)$ to obtain a noisy version of H.

The noise injection in Step 3 is justified by the following theorem.

Theorem 3 ([13]) *Let $\Delta_2(H)$ denote the L_2-sensitivity of a histogram H, and let $\Delta_1(\boldsymbol{F}^k)$ denote the L_1-sensitivity of its first k DFT coefficients $\boldsymbol{F}^k$. Then, $\Delta_1(\boldsymbol{F}^k) \leq \sqrt{k}\Delta_2(H)$.* ■

FPA provably satisfies ε-differential privacy [13]. We denote the absolute error of a single bin H_i in H by $\mathbb{E}|H_i - \hat{H}_i|$ and the reconstruction error due to ignoring $n-k$ coefficients by $RE^k_i(H_i)$. Theorem 4 quantifies the absolute error of $\hat{H}_i$.

Algorithm 1 Sampling Perturbation (SPA)

Input: Raw histogram H with length n, where n is odd
Input: Privacy budget ε
Output: Noisy histogram $\hat{H}$ with length n

1: $\mathbf{F} := \text{DFT}(H)$
2: $\forall 1 \leq k \leq n$, compute $U(H,k) = \sqrt{\sum_{i=k+1}^{n} |F_{i-1}|^2} + \frac{k\sqrt{n}}{\varepsilon}$
3: Select k with probability $\propto \exp\left(-\frac{\varepsilon \cdot U(H,k)}{\sqrt{2}}\right)$
4: $g := \mathcal{G}(\varepsilon^2/2, (k+1)/2)$
5: $\hat{\mathbf{F}}^k := \mathbf{F}^k + \langle \mathcal{N}(0, \sqrt{g}) \rangle^k$
6: **return** $\hat{H} = \text{IDFT}(\text{PAD}^n(\hat{\mathbf{F}}^k))$

Theorem 4 ([13]) $\forall 1 \leq i \leq n$, $\mathbb{E}|H_i - \hat{H}_i| \leq RE^k_i(H_i) + \frac{k}{n\varepsilon}$. ■

3) Sampling Perturbation Algorithm (SPA): In FPA, the number of coefficients to retain, k, in Step 2 is given as an input. However, the choice of a good value of k is critical to the utility of FPA. Theorem 4 shows that the total error is composed of two parts: the perturbation error $\frac{k}{n\varepsilon}$ due to adding Laplace noise to $\mathbf{F}^k$ and the reconstruction error $RE^k_i(H_i)$ due to ignoring $n-k$ DFT coefficients. There is a fundamental trade-off between the perturbation error and the reconstruction error: when k is larger, perturbation error increases while reconstruction error decreases; when k is smaller, these two types of errors behave oppositely. This fact suggests that k has to be carefully selected so that the total error can be minimized.

Rastogi and Nath [13] also propose an extension to FPA, called Sampling Perturbation Algorithm (SPA), to select k adaptively depending on a dataset. SPA (given in Algorithm 1) uses exponential mechanism to select a value of k by the utility function $-U(H,k)$, where

$$U(H,k) = \left(\sum_{i=k+1}^{n} |F_{i-1}|^2 \right)^{\frac{1}{2}} + \frac{k\sqrt{n}}{\varepsilon}. \tag{1}$$

SPA samples both k and the perturbed $\hat{\mathbf{F}}^k$ using exponential mechanism. The sampling of $\hat{\mathbf{F}}^k$ from a multidimensional hyperbolic distribution is done by first sampling g from a gamma distribution $\mathcal{G}(\varepsilon^2/2, (k+1)/2)$, and then perturbing $\mathbf{F}^k$ with i.i.d Gaussian noise $\mathcal{N}(0, \sqrt{g})$.

B. Our Proposal: Enhanced Fourier Perturbation Algorithm

SPA is sub-optimal due to at least two reasons: 1) the utility function $-U(H,k)$ overestimates the perturbation error; 2) SPA disregards the fact that the coefficients of a real-valued histogram are correlated. In the rest of this section, we discuss these two issues in more details and design a more effective perturbation scheme.

Utility function design. We reconsider the problem of selecting the first k coefficients from the perspective of *denoising*

in statistical signal processing [2]. We first precisely quantify the *sum of squared error* (SSE) of FPA.

Lemma 1 *Given* $\hat{H} = FPA(H, \varepsilon)$, $\mathbb{E}||H - \hat{H}||_2^2 = \sum_{i=k+1}^{n} |F_{i-1}|^2 + \frac{2k^2}{\varepsilon^2}$. ■

Proof: Since DFT is an orthonormal transformation, it preserves the L_2-norm. Hence, $||H - \hat{H}||_2 = ||\mathbf{F} - \hat{\mathbf{F}}||_2$, and therefore the sum of squared error of $\hat{H}$ is $\mathbb{E}||H - \hat{H}||_2^2 = \mathbb{E}||\mathbf{F} - \hat{\mathbf{F}}||_2^2$. For the first k coefficients in $\hat{\mathbf{F}}$, we have $\hat{F}_i = F_i + \mathcal{L}(\sqrt{k}/\varepsilon)$ $(i \leq k)$. Since $\mathcal{L}(\sqrt{k}/\varepsilon)$ is independently added to both the real and imaginary part of F_i, the squared error is $2 \cdot \mathbb{E}[\mathcal{L}(\sqrt{k}/\varepsilon)^2] = 4k/\varepsilon^2$. For the rest $n-k$ coefficients, the error is $|F_{i-1}|^2$ $(i > k)$ due to the removal of these coefficients. Therefore, the SSE of $\hat{H}$ is

$$\begin{aligned} \mathbb{E}||H - \hat{H}||_2^2 = \mathbb{E}||\mathbf{F} - \hat{\mathbf{F}}||_2^2 &= \sum_{i=1}^{k} \frac{4k}{\varepsilon^2} + \sum_{i=k+1}^{n} |F_{i-1}|^2 \\ &= \sum_{i=k+1}^{n} |F_{i-1}|^2 + \frac{4k^2}{\varepsilon^2} \quad (2) \end{aligned}$$

■

Again, Lemma 1 indicates that the total error consists of the perturbation error $\frac{4k^2}{\varepsilon^2}$ due to adding Laplace noise to k DFT coefficients and the reconstruction error $\sum_{i=k+1}^{n} |F_{i-1}|^2$ due to ignoring $n-k$ DFT coefficients. Intuitively, we can observe that the utility function $-U(H, k)$ used by SPA overestimates the perturbation error. If LPA is used to add noise to the first k coefficients, the perturbation error can be much less than $k\sqrt{n}/\varepsilon$.

Next, under this observation, we design a better utility function than $-U(H, k)$, which more precisely estimates the true root-sum-squared error (RSSE), if we perturb the coefficients by LPA.

Theorem 5 *For FPA,* $\mathbb{E}||H - \hat{H}||_2 \leq \sqrt{\sum_{i=k+1}^{n} |F_{i-1}|^2} + \frac{2k}{\varepsilon}$.

Proof:

$$\begin{aligned} \mathbb{E}||H - \hat{H}||_2 &= \mathbb{E}\left(\sqrt{||H - \hat{H}||_2^2}\right) \\ &\leq \sqrt{\mathbb{E}||H - \hat{H}||_2^2} \quad \text{(by Jensens' inequality)} \\ &= \sqrt{\sum_{i=k+1}^{n} |F_{i-1}|^2 + \frac{4k^2}{\varepsilon^2}} \quad \text{(by Lemma 1)} \\ &\leq \sqrt{\sum_{i=k+1}^{n} |F_{i-1}|^2} + \sqrt{\frac{4k^2}{\varepsilon^2}} \end{aligned}$$

where the last inequality is due to $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$. This concludes the theorem. ■

Based on Theorem 5, we design our new utility function as $-u(H, k)$, where

$$u(H, k) = \sqrt{\sum_{i=k+1}^{n} |F_{i-1}|^2} + \frac{2k}{\varepsilon}. \quad (3)$$

Algorithm 2 Enhanced Fourier Perturbation (EFPA)

Input: Raw histogram H with length n, where n is odd
Input: Privacy budget ε
Output: Noisy histogram $\hat{H}$ with length n

1: $\mathbf{F} := \text{DFT}^{\mathsf{real}}(H)$
2: $m := |\mathbf{F}| = (n+1)/2$
3: Compute $u(H, k) = \sqrt{\sum_{i=k+1}^{m} 2|F_{i-1}|^2} + \frac{2z}{\varepsilon}$ for all $1 \leq k \leq m$, where $z = 2k + 1$
4: Select k with probability $\propto \exp\left(-\frac{\varepsilon \cdot u(H,k)}{4}\right)$
5: $z := 2k + 1$
6: $\hat{\mathbf{F}}^k := \mathbf{F}^k + \langle \mathcal{L}(2\sqrt{z}/\varepsilon) \rangle^k$
7: **return** $\hat{H} = \text{IDFT}^{\mathsf{real}}(\text{PAD}^m(\hat{\mathbf{F}}^k))$

Using $-u(H, k)$, exponential mechanism will favor a k value that results in less RSSE. Theorem 5 shows that using $-u(H, k)$ to select k and perturbing the first k retained coefficients by LPA (instead of using exponential mechanism as SPA does) are expected to result in lower $\mathbb{E}||H - \hat{H}||_2$ than SPA.

The choice of using RSSE in $-u(H, k)$ and $-U(H, k)$ deserves more explanation: although we could directly use SSE, $\sum_{i=k+1}^{n} |F_{i-1}|^2 + \frac{4k^2}{\varepsilon^2}$, as the utility function to select k, the sensitivity of $\sum_{i=k+1}^{n} |F_{i-1}|^2$ can be much larger than that of RSSE. The sensitivity of $\sum_{i=k+1}^{n} |F_{i-1}|^2$ is actually data-dependent, since it is determined by $\max |F_i|$. This can result in very poor results if $\max H_i$ is large. By contrast, the sensitivity of $\sqrt{\sum_{i=k+1}^{n} |F_{i-1}|^2}$ is bounded by 1.

Theorem 6 $\Delta_1 u(H, k) = 1$ ■

Proof: $\Delta_1 u(H, k) = \Delta_1 \left(\sqrt{\sum_{i=k+1}^{n} |F_{i-1}|^2}\right) \leq \Delta_1(||\mathbf{F}||_2)$. Consider two neighboring histograms H and H', where $\text{DFT}(H) = \mathbf{F}$ and $\text{DFT}(H') = \mathbf{F}'$. Since the L_2-norm (i.e., RSSE) is invariant under DFT, we obtain $||H||_2 = ||\mathbf{F}||_2$ and $||H'||_2 = ||\mathbf{F}'||_2$. Hence, $\big|\, ||\mathbf{F}||_2 - ||\mathbf{F}'||_2 \,\big| = \big|\, ||H||_2 - ||H'||_2 \,\big| \leq ||H - H'||_2 \leq ||H - H'||_1 \leq 1$ due to the reverse triangle inequality and the standard inequality between norms. ■

Figure 2 illustrates the utility improvement of EFPA over SPA. since SPA consistently overestimates the perturbation error, it always retains and perturbs fewer DFT coefficients, leading to larger reconstruction error, whereas EFPA finds a better trade-off between these errors, and therefore more faithfully preserves the general shape of the original histogram.

Real DFT vs. Complex DFT. Both basic FPA and SPA consider all DFT coefficients. However, the coefficients of real-valued histograms are correlated as they are half-redundant (i.e., $\hat{F}_i = \hat{F}^*_{n-i}$, for $1 \leq i \leq (n-1)/2$). SPA does not exploit this correlation and defines $-U(H, k)$ over all coefficients. This not only results in a complex-valued histogram, but also accumulates the Laplace noise added to each coefficient on the bins, making the scheme less effective. As a result, we apply *real* DFT (i.e., $\text{DFT}^{\mathsf{real}}$ described in Section IV-A1) on the real-valued input histogram, and perturb the resulted coefficients

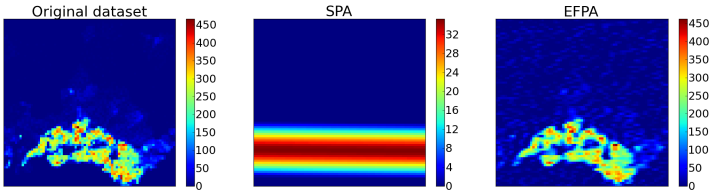


Fig. 2. SPA vs. EFPA ($\varepsilon = 0.1$) on a location map (*Location* dataset in Section VI) consisting of a grid with 88 $\times$ 87 cells. Each cell represents a bin and is colored based on its (noisy) count.

with LPA, where the coefficients are selected by exponential mechanism using the score function $-u(H, k)$.

Our new algorithm, called enhanced FPA (EFPA), is summarized in Algorithm 2. To ease presentation, we assume n to be odd. As EFPA applies DFT^{real} on H, it operates on m instead of n coefficients. k is selected in Line 4 by exponential mechanism with budget $\varepsilon/2$. Then, EFPA perturbs each F_i^k $(0 \leq i \leq k-1)$ with $\mathcal{L}(2\sqrt{z}/\varepsilon)$ in Line 6 (the real-valued noise is only added to the magnitude of coefficients). Note that the number of retained coefficients is $z = 2k+1$ because except F_0^k the complex conjugate of all other coefficients also appears in the complete Fourier transform of H (see Section IV-A1). Finally, $\hat{\mathbf{F}}^k$ is padded to a m-dimensional vector by appending $m-k$ zeros, denoted by $\text{PAD}^m(\hat{\mathbf{F}}^k)$, and the inverse DFT^{real} is applied to obtain $\hat{H}$. The total budget ε is uniformly divided between LPA (Line 6) and exponential mechanism (Line 4), therefore, EFPA is ε-differentially private due to the *sequential composition* property [11].

V. Clustering-based Sanitization

We start by giving the intuition why clustering can be used for boosting accuracy. Consider a simple strategy: we first cluster bins in H that have close values, and then replace the count of each bin by the mean value of the cluster to which it belongs (i.e., the cluster center) plus properly calibrated Laplace noise. This strategy results in a smaller sensitivity and therefore less perturbation error. If the counts within each cluster are close enough (hence small reconstruction error), this technique can lower the total error in the released histogram (see Theorem 7 for a formal statement).

In particular, given k clusters $\mathbf{C} = \{C_1, \ldots, C_k\}$ over H's bins. We call $\mathbf{C}$ a *cluster configuration* of bins. Ideally, one wants to form clusters such that the average distance between each count and its cluster center, i.e., $\sum_{i=1}^{k} \sum_{H_j \in C_i} |H_j - \overline{C}_i|$ is minimized, where $\overline{C}_i = \sum_{H_j \in C_i} H_j/|C_i|$. Then, we can obtain an approximation of H, denoted by $H_{\mathbf{C}}$, by replacing each H_i in H with its cluster center. Now, let us assume for the moment that we can form these clusters without violating privacy. Having the approximated histogram $H_{\mathbf{C}}$, we add i.i.d Laplace noise to each cluster center $\{\overline{C}_1 + \frac{\mathcal{L}(1/\varepsilon)}{|C_1|}, \ldots, \overline{C}_k + \frac{\mathcal{L}(1/\varepsilon)}{|C_k|}\}$, and finally release the noisy approximation $\hat{H}_{\mathbf{C}}$ of H, which is obtained by replacing each H_i in H with its noisy cluster center. This idea is formalized in Algorithm 3.

We show that if a cluster configuration $\mathbf{C}$ is properly selected (i.e., the bins within each C_i have close counts), we can obtain better utility than LPA. Let $err(\hat{H}_i)$ denote the absolute error of the noisy count $\hat{H}_i$ of bin i in $\hat{H}_{\mathbf{C}}$: $err(\hat{H}_i) = \mathbb{E}|H_i - \hat{H}_i|$, and $err(\hat{H}_{\mathbf{C}}) = \sum_{i=1}^{n} err(\hat{H}_i)$. Similarly, we denote the histogram generated by LPA by $\widetilde{H}$.

Algorithm 3 *ReleaseHistogram*

Input: Raw histogram H
Input: Privacy budget ε
Input: Cluster configuration $\mathbf{C} = \{C_1, \ldots, C_k\}$ of H
Output: Noisy histogram $\hat{H}_{\mathbf{C}}$

1: $\overline{C}_i := \sum_{H_j \in C_i} H_j/|C_i|$
2: $\mathbf{U} := \{\overline{C}_1 + \frac{\mathcal{L}(1/\varepsilon)}{|C_1|}, \ldots, \overline{C}_k + \frac{\mathcal{L}(1/\varepsilon)}{|C_k|}\}$
3: $\hat{H}_{\mathbf{C}} = \{\overbrace{\underbrace{U_1, \ldots, U_1}_{|C_1|}, \underbrace{U_2, \ldots, U_2}_{|C_2|}, \ldots, \underbrace{U_k, \ldots, U_k}_{|C_k|}}^{n}\}$
4: **return** $\hat{H}_{\mathbf{C}}$

Theorem 7 *Let $\hat{H}_{\mathbf{C}}$ denote a noisy approximation of H as detailed above. Then, $err(\hat{H}_{\mathbf{C}}) \leq RE_{H_{\mathbf{C}}} + k/\varepsilon$, in contrast to $err(\widetilde{H}) = n/\varepsilon$.* ■

Proof: The error of $\widetilde{H}$ is $\mathbb{E}\left(\sum_{i=1}^{n} |H_i - \widetilde{H}_i|\right) = n/\varepsilon$. The error of $\hat{H}_{\mathbf{C}}$ can be calculated as follow:

$$\begin{aligned}
\mathbb{E}\left(\sum_{i=1}^{n} |H_i - \hat{H}_i|\right) &= \sum_{i=1}^{k} \sum_{H_j \in C_i} \mathbb{E}\left|H_j - \overline{C}_i - \frac{\mathcal{L}(1/\varepsilon)}{|C_i|}\right| \\
&\leq \sum_{i=1}^{k} \sum_{H_j \in C_i} \mathbb{E}|H_j - \overline{C}_i| + \\
&\quad \sum_{i=1}^{k} \sum_{H_j \in C_i} \frac{\mathbb{E}|\mathcal{L}(1/\varepsilon)|}{|C_i|} \\
&= \sum_{i=1}^{k} \sum_{H_j \in C_i} |H_j - \overline{C}_i| + k/\varepsilon
\end{aligned}$$

The reconstruction error $RE_{H_{\mathbf{C}}} = \sum_{i=1}^{k} \sum_{H_j \in C_i} |H_j - \overline{C}_i|$. ■

In the extreme case, where all counts in a cluster C_i have the same value, $err(\hat{H}_j) = \frac{1}{|C_i|} err(\widetilde{H}_j)$. For the entire histogram H, our intuition is that if the number of clusters $k \ll n$ and the reconstruction error $RE_{H_{\mathbf{C}}}$ is small, $\hat{H}_{\mathbf{C}}$ will be much closer to H than $\widetilde{H}$.

Slightly abusing the notation, let $err(\hat{H}_{\mathbf{C}})$ denote $RE_{H_{\mathbf{C}}} + k/\varepsilon$ in the sequel, and to simplify the notation we denote $err(\hat{H}_{\mathbf{C}})$ by $err(\mathbf{C}, \varepsilon)$ if it is unambiguous in the context. Note that $err(\hat{H}_{\mathbf{C}})$ has two parts: 1) the *reconstruction error* $RE_{H_{\mathbf{C}}}$ due to the approximation of each bin count with its cluster center, and (2) the *perturbation error* k/ε due to the injected Laplace noise. There is a trade-off between them: creating more clusters decreases the reconstruction error at the cost of larger perturbation error. Clearly, there exists an optimal clustering configuration such that the sum of the two errors is minimal among all possible configurations for a given H and ε.

Finding the optimal configuration is *NP-hard*. Instead, our goal is to find a sub-optimal configuration without vio-

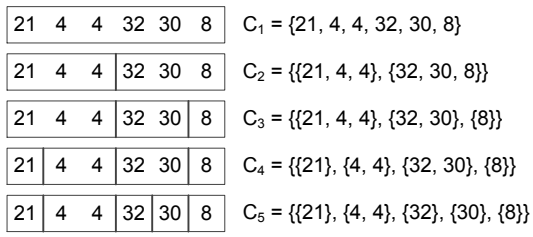


Fig. 3. Operation of *HPartition*.

lating differential privacy. Indeed, we have omitted so far the fact that clustering must also be differentially private. Therefore, given the total privacy budget ε, we design a $\frac{\varepsilon}{2}$-differentially private clustering algorithm to compute a configuration $\mathbf{C}$, which minimizes $err(\mathbf{C}, \frac{\varepsilon}{2})$, the error of $\hat{H}_{\mathbf{C}} = \textit{ReleaseHistogram}(\mathbf{C}, \frac{\varepsilon}{2})$.

A. HPartition: Hierarchical Partitioning of Histogram Bins

Our private clustering scheme is based on an incremental partitioning algorithm called *HPartition*. *HPartition* finds the sub-optimal partitions (clusters) by taking into consideration two types of costs (i.e., errors) incurred in the partition process. The first cost corresponds to the reconstruction error as defined previously. It decreases as the number of partitions increases. The second cost is quantified by perturbation error: creating a new partition implies more Laplace noise to be added. We assume for now that this cost is fixed and equals λ for each new partition.

HPartition splits a histogram $H = \{H_1, H_2, \ldots, H_n\}$ into k partitions, $\mathbf{C}_k = \{C_1^k, C_2^k, \ldots, C_k^k\}$, such that $err(\mathbf{C}_k) = RE_{\mathbf{C}_k} + k\lambda$ is minimized, where $RE_{\mathbf{C}_k} = \sum_{i=1}^{k} \sum_{H_j \in C_i} |H_j - \overline{C}_i|$. At the beginning, *HPartition* considers the configuration with a single partition containing all n bins, that is, $\mathbf{C}_1 = \{C_1^1 = \{H_1, H_2, \ldots, H_n\}\}$. For each of $n-1$ possible bisections $B_i(C_1^1) = \{\{H_1, \ldots, H_i\}, \{H_{i+1}, \ldots, H_n\}\}$, which divides C_1^1 into two sub-partitions, it computes its error $e_2 = err(B_i(C_1^1))$ (i.e., the cost of conducting B_i). It then selects the bisection $B_k(C_1^1)$ with the minimum error e_2 among all $n-1$ possible bisections. If this error is smaller than $e_1 = err(\mathbf{C}_1)$, the algorithm continues and sets $\mathbf{C}_2 = \{C_1^2 = \{H_1, \ldots, H_k\}, C_2^2 = \{H_{k+1}, \ldots, H_n\}\}$ (otherwise, it stops and returns $\mathbf{C}_1$). In the next iteration, *HPartition* considers all possible bisections on C_1^2 and C_2^2 and computes their resultant errors, $e_3 = err(\{B_i(C_1^2), C_2^2\})$ and $e_3 = err(\{C_1^2, B_i(C_2^2)\})$, respectively. It picks the minimal e_3 and compares it with $err(\mathbf{C}_2)$. If e_3 is smaller than $err(\mathbf{C}_2)$, the algorithm continues, else it stops and returns $\mathbf{C}_2$. The algorithm terminates when the partition process cannot decrease the total error any more. *HPartition* is a greedy heuristic: in each step, it always selects the bisection that results in the greatest error improvement.

Example 1 *Figure 3 illustrates the operation of* HPartition *on* $H = \{21, 4, 4, 32, 30, 8\}$. *Suppose that* $\lambda = 1$. *For* $\mathbf{C}_4$, *there are two partitions to bisect which are* $C_2^4 = \{4, 4\}$ *and* $C_3^4 = \{32, 30\}$. *Bisecting* C_3^4 *lowers* $RE_{\mathbf{C}_4}$ *by 2, and decreases the total cost by 1. By contrast, bisecting* C_2^4 *does not decrease* $RE_{\mathbf{C}_4}$, *but increases the total cost by 1 due to the introduction of a new partition. Hence,* HPartition *bisects* C_3^4 *and releases* $\mathbf{C}_5$. *However, if* $\lambda = 3$, *bisecting* C_3^4 *would increase the total error, and therefore* HPartition *releases* $\mathbf{C}_4$ *and stops.* ■

Algorithm 4 *P-HPartition*

Input: Raw histogram $H = \{H_1, H_2, \ldots, H_n\}$
Input: Privacy budget ε
Output: Noisy approximated histogram $\hat{H}_{\mathbf{C}}$

Create a queue $\mathbf{C}$ with a single partition $\{H_1, H_2, \ldots, H_n\}$
Mark the partition in $\mathbf{C}$ as non-leaf
$\mathbb{C} := \{\mathbf{C}\}$
while there exists a non-leaf partition in $\mathbf{C}$ **do**
  $C :=$ the first non-leaf partition in $\mathbf{C}$
  `//Bisect the selected cluster at all positions`
  $P_i := B_i(C)$ for all $0 \le i \le |C| - 1$
  $\mathbf{C}_i := (\mathbf{C} \backslash C) \cup P_i$
  Select P_k with probability $\propto \exp(-\frac{\varepsilon \cdot err(\mathbf{C}_k, \varepsilon/2)}{16 \cdot \lfloor \log_2 n \rfloor})$
  if $k = 0$ **then** `//if no bisection is chosen`
    Mark C as leaf
  `//Consider all resultant sub-partitions`
  for all $c \in P_k$ **do**
    if $|c| = 1$ **or** c has been bisected $\lfloor \log_2 n \rfloor$ times
      Mark c as leaf
  end for
  Remove C from $\mathbf{C}$
  Append C's children P_k to $\mathbf{C}$
  $\mathbb{C} := \mathbb{C} \cup \mathbf{C}$ `//save the new configuration`
end while
Select $\mathbf{C}_{final} \in \mathbb{C}$ with prob. $\propto \exp(-\frac{\varepsilon \cdot err(\mathbf{C}_{final}, \varepsilon/2)}{16})$
return *ReleaseHistogram*$(\mathbf{C}_{final}, \varepsilon/2)$

B. P-HPartition: Private Partitioning of Histogram Bins

We make *HPartition* differentially private by using exponential mechanism. The new algorithm, called *P-HPartition*, is described in Algorithm 4. *HPartition* has two data-dependent parts: 1) the selection of the best bisection in each iteration; 2) the decision whether there exists a bisection in a single iteration that leads to smaller error.

The first part would become private by using exponential mechanism to select from all possible bisections, where the score (utility) function is inversely proportional to the error of the configuration produced by a bisection. However, in that case, the privacy budget has to be divided among *all* possible bisections, whose number can be as large as n (i.e., the number of bins), since in the worst case a single record difference influences all bisections. If a histogram has a large number of bins, this approach would quickly become unusable because the small privacy parameter makes the selection of a bisection almost random.

To tackle this challenge, we leverage on a tree structure of the clustering hierarchy. The initial partition containing all n bins forms the root of the tree; each bisection splits a partition (represented as a node in the tree) into two child nodes. Each bisection consists of two steps: 1) select a partition who has not been bisected for more than d times before, which is made data-independent and therefore does not violate differential privacy; 2) bisect the selected partition using exponential

mechanism. Since the clustering tree structure is a *binary tree*, we heuristically set $d = \lfloor \log_2 n \rfloor$, which performs stably well in all experiments. This approach results in better utility, because the privacy budget will be divided among $\lfloor \log_2 n \rfloor$ bisections as opposed to n.

We have to make sure that the selection of a partition to bisect is data-independent (i.e., we do not prioritize any partition). For this purpose, we maintain a queue of all partitions, and always pick up the first partition in the queue to bisect. After each bisection, the resultant sub-partitions are appended to the end of the queue, and the new configuration (i.e., all partitions in the queue) is saved. The partition process terminates, if no partition can be bisected any more because 1) they are singletons or, (2) they have already been bisected for d times or, (3) the bisection would result in larger error (note that the exponential mechanism we design can select not to bisect). When the partition process ends, the final configuration is selected among the saved ones using a second exponential mechanism. The utility (score) function of both exponential mechanisms is $-err(\hat{H}_{\mathbf{C}})$, which is inversely proportional to the error of the configuration $\mathbf{C}$.

Lemma 2 $\Delta err \leq 2$. ■

Proof: Since $err(\hat{H}_{\mathbf{C}}) = RE_{H_{\mathbf{C}}} + k/\varepsilon$, the sensitivity of err is determined by the sensitivity of $RE_{H_{\mathbf{C}}} = \sum_{i=1}^{k} \sum_{H_j \in C_i} |H_j - \overline{C}_i|$. Consider a partition C with counts $\{c_1, c_2, \ldots, c_n\}$ and let C' differ from C only in a single bin by 1. Then, $\sum_i |c'_i - \frac{\sum_i c'_i}{n}| \leq \sum_i |c_i - \frac{\sum_i c_i}{n}| + 2$. ■

Theorem 8 (Privacy) *P-HPartition satisfies ε-differential privacy.* ■

Proof: Following from Lemma 2 and Theorem 2, a single bisection is $\varepsilon/4d$-differentially private. Since a single record difference in H affects at most d bisections, the entire partition process is $\varepsilon/4$-differentially private. Similarly, the selection of the best configuration (Line 21) is also $\varepsilon/4$ private. Moreover, since *ReleaseHistogram* is $\varepsilon/2$-differentially private, the whole scheme is ε-differentially private due to the sequential composition property of differential privacy [11]. ■

Recall that the cost of adding a new cluster is $\lambda = 2/\varepsilon$.

C. Design Choices of P-HPartition

Another private clustering scheme has been recently proposed in [16]. It proposes a differentially private version of the optimal partitioning scheme in [8] to create clusters. Two strategies are introduced: *StructureFirst* uses exponential mechanism to randomize the partitioning process and then adds noise, whereas *NoiseFirst* first adds Laplace noise and then performs clustering based on noisy counts. These schemes suffer from at least three drawbacks: 1) they use a simple heuristic to compute the number of clusters (i.e., $k = |H|/10$), which disregards the essential trade-off between the reconstruction and perturbation errors. In contrast, our *P-HPartition* scheme adaptively decides whether to create a new cluster by taking into consideration both injected Laplace noise and reconstruction error, leading to better performance. 2) the reconstruction error is measured by the sum of squared error (SSE) in [16] (as opposed to the sum of absolute error used in this paper), whose sensitivity is upper bounded by the greatest count in a histogram. Such a data-dependent upper bound can result in very inaccurate perturbation in case of large counts. In contrast, as shown in Lemma 2, Δerr is upper bounded by 2, independent of the underlying counts. 3) the schemes in [16] are inefficient for large histograms as the complexity of their clustering schemes are $O(kn^2)$, where k is the number of clusters. By contrast, *HPartition* (or *P-HPartition*) has a worst case complexity of $O(n^2)$ [1]. This divisible technique also ensures that the error only increases as a logarithmic function of n.

TABLE I
EXPERIMENTAL DATASET CHARACTERISTICS.

Dataset	|H|	Number of zero count	Mean	Variance	Count Range
Location	7,725	4,720	24.13	4,764.57	[0, 467]
Rochdale	256	165	2.60	52.12	[0, 57]
Search Log	32,768	17,082	10.25	577.31	[0, 496]
NetTrace	65,536	63,318	0.39	91.01	[0, 1,423]

VI. EXPERIMENTAL EVALUATION

In this section, we experimentally evaluate the utility of the two solutions we propose, namely *P-HPartition* (P-HP) and *EFPA*. As a reference point, we compare our solutions with six state-of-the-art methods: *LPA* [3], *SPA* [13], *Privelet* [14], *Boost* [7], *MWEM* [5] and the clustering schemes from [16]. We examine the utility of different approaches under both unattributed and attributed histograms. Our implementation was done in Python, and all experiments were performed on an Intel Xeon 2.27GHz PC with 8GB RAM. The source code and the datasets used are available at http://planete.inrialpes.fr/projects/p-publication/.

A. Datasets

Four datasets (both real-life and synthetic) with different characteristics are used in our experiments. *Location* is a half-synthetic dataset generated from the 2006 Census Meshblock Dataset[2] of New Zealand. We randomly selected an administrative region of New Zealand, called Waitakere, with a total population of 186,471 distributed in 1,340 meshblocks [3]. We produced the dataset by randomly placing the residents into each meshblock and then dividing the whole area into 7,725 non-overlapping rectangles (of size 154×113 m^2), each being a bin in the histogram. The bin count is the number of residents covered by its rectangle. *Rochdale* [4] is a 2-dimensional histogram having 16×16 bins. It relates to women's economic activity and husbands' unemployment from a household survey in Rochdale, Lancashire, UK. *Search Log* [7] is a synthetic dataset of search query logs for the

[1] It compares at most $n-1$ bisections in each step, and there are at most n steps. Since the error computation in each iteration takes $O(n)$, the total complexity is $O(n + 2n^2) = O(n^2)$.

[2] http://www.stats.govt.nz/Census/2006CensusHomePage/MeshblockDataset.aspx

[3] A meshblock is the smallest data unit, each of which covers an area of around 30-60 dwellings.

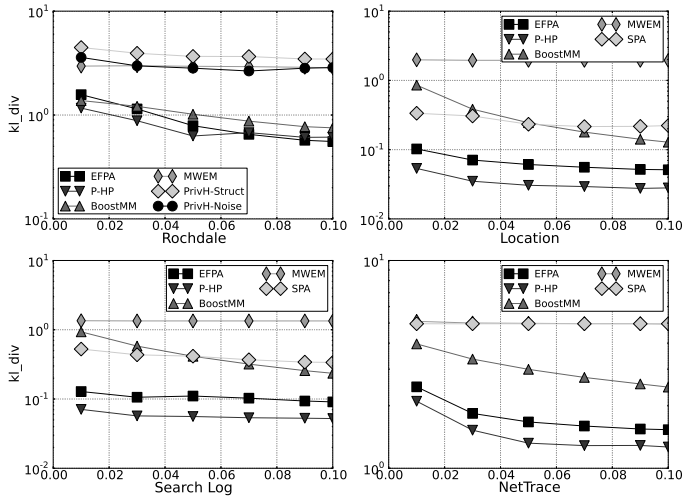


Fig. 4. Unattributed histograms: KL-divergence under varying ε values. y-axis is in log-scale.

keyword "Obama" issued from Jan. 1, 2004 to Aug. 9, 2009. It contains 32,768 bins, each giving the number of queries issued within a 90-minutes interval. *NetTrace* [7] describes the IP-level network traces collected at a university. There are 65,536 bins, each giving the number of external hosts that are connected to an internal host. The detailed characteristics are summarized in Table I. Note that since *Location* and *Rochdale* are 2-dimensional histograms, we converted them into 1-dimensional histograms by concatenating consecutive rows.

B. Unattributed Histograms

The main purpose of publishing an unattributed histogram is to study its probability distribution and to answer different aggregate queries, which do not require the associations between bins and counts. In particular, we examine the usefulness of released unattributed histograms in terms of KL-divergence.

Besides SPA [13], we compare our solutions to the following schemes: *BoostMM* [7], [6], which is a post-processing mechanism based on least squares regression over ordered histograms, *MWEM* [5], which is based on an iterative constructive mechanism using the multiplicative weight approach and, to the best of our knowledge, has the best worst case error bound, and the clustering-based algorithms in [16], which have been described in Section V-C. *NoiseFirst* and *StructureFirst* are denoted by *PrivH-Noise* and *PrivH-Struct*, respectively, in the sequel. Since *PrivH-Noise* and *PrivH-Struct* are less scalable to large histograms [4], we report their performance only on the *Rochdale* dataset. In all following figures, the reported results are the average of 100 runs.

Figure 4 reports the KL-divergence of different datasets under varying ε values (from 0.01 to 0.1). We observe that, for unattributed histograms, *P-HPartition* outperforms the other approaches for all datasets. We also observe that *EFPA* significantly outperforms *SPA*, which justifies our optimization. It can be seen that *P-HPartition* achieves the largest utility gain on large histograms (*Location*, *SearchLog* and *NetTrace*). This is because many counts in these histograms have similar values, which favours clustering (see Theorem 7).

[4]A single experiment with *PrivH-Noise* or *PrivH-Struct* on any of our datasets, except *Rochdale*, takes several days to complete on our machine. We used the code available on the authors' site: `http://faculty.neu.edu.cn/ise/xujia/home/DP-Introduction.html`

TABLE II
KL-DIVERGENCE OF ATTRIBUTED HISTOGRAMS ($\varepsilon = 0.01$)

Dataset	EFPA	P-HP	SPA	LPA	MWEM
Location	0.57	1.01	1.17	1.40	1.98
Search Log	0.18	0.27	0.49	2.30	1.35
NetTrace	2.49	1.78	4.96	5.09	5.11

TABLE III
KL-DIVERGENCE OF ATTRIBUTED HISTOGRAMS ($\varepsilon = 0.01$)

Dataset	EFPA	P-HP	MWEM	PrivH-Struct	PrivH-Noise
Rochdale	1.76	2.23	3.02	4.47	3.14

C. Attributed Histograms

We first examine the usefulness of sanitized attributed histograms in terms of KL-divergence in Table II and III. Since the histograms are unordered, we omit *BoostMM* and add *LPA* to our evaluation. Similarly, *PrivH-Struct* and *PrivH-Noise* are not scalable to large datasets. Due to the space limit, we only report the results for $\varepsilon = 0.01$. Similar trends can be observed when $\varepsilon = 0.1$. We observe that *EFPA* and *P-HPartition* achieve smaller divergence. In particular *EFPA* performs the best, except when the dataset is sparse (e.g., *NetTrace*). In this case, *P-HPartition* is the best scheme.

One of the most common data analysis task conducted on attributed histograms is range count queries [14], [7]. We follow the convention to evaluate the utility of sanitized histograms in terms of *mean squared error* (MSE) (see Section III-C). Similar to [7], the range sizes are 2^i for $i = 1, \ldots, \lfloor \log n \rfloor$, where n is the number of bins in each dataset. We compute all queries for a given size, and report their average error over 100 runs. We compare our approaches with all previous schemes as well as *Privelet* [14] and *BoostTree* [7], where the last one is a linear unbiased estimator of range queries using their noisy values on universal (unordered) histograms.

Figure 5 studies how MSE varies under different range sizes. It can be observed that our approaches outperform the other schemes in most cases, and in some datasets the improvement is up to a factor of 10. For the reason mentioned before, on *NetTrace*, *P-HPartition* performs better than *EFPA*. As ε becomes smaller (i.e., privacy increases), the difference between our schemes and the other schemes increases. Another major trend observed in Figure 5 is that in general MSE increases with the increment of range size, because normally larger ranges accumulate more noise. It is interesting to note that *SPA* and *EFPA* perform similarly on *NetTrace*. This is because *NetTrace* is almost ordered, and hence the signal ("histogram") energy is concentrated on low-frequency components based on Parseval's theorem. Since *SPA* overestimates the perturbation error, it always keeps the low-frequency components, which is coincidentally the correct approach on *NetTrace*. However, when a histogram is not ordered, which is the usual case,

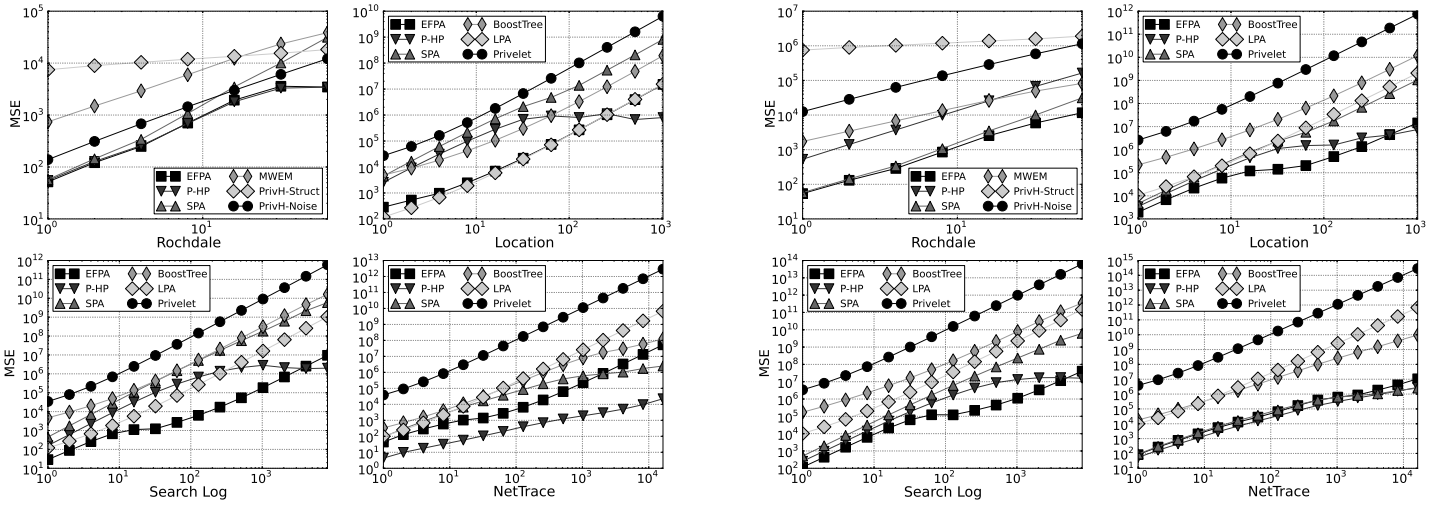


Fig. 5. Attributed histograms: MSE on different datasets under varying range sizes. y-axis is in log-scale. Left: $\varepsilon = 0.1$ Right: $\varepsilon = 0.01$

EFPA significantly outperforms *SPA* as demonstrated by the results on other datasets.

VII. Discussion

As suggested by the experimental results, our schemes find a better trade-off between the reconstruction and perturbation errors under the idea of lossy compression. While our schemes outperform the existing solutions, the improvement depends on the compressibility of a histogram. Generally, histograms that have more similar counts provide higher compressibility, and favour our schemes. For this reason, our techniques are especially well-suited to unattributed histograms, whose compressibility can be significantly increased by ordering (similar counts are then in neighboring positions). This property makes *EFPA* and *P-HPartition* very accurate sanitization techniques for unattributed histograms: for *EFPA*, ordering shifts the signal energy from high-frequency to low-frequency components based on Parseval's theorem, and therefore allows the low-pass filter used by *EFPA* to select the best trade-off between the reconstruction and perturbation errors; for *P-HPartition*, ordering allows to form larger clusters with similar counts, and therefore reduces the noise. Nevertheless, many real-life datasets (e.g., location datasets) are already partially ordered and sparse. Hence, their compressibility, even without explicit ordering, is high enough for our schemes to achieve good utility.

VIII. Conclusion

We have presented two novel sanitization algorithms for generating differentially private histograms. Our schemes are based on the general idea of lossy compression. The first scheme optimizes the Fourier Perturbation Algorithm by more rigorous utility analysis while the second scheme makes use of a divisible hierarchical clustering algorithm. Since many real-life datasets are inherently highly-compressible, our schemes exhibit great promise for outperforming the state-of-the-art solutions. This is further confirmed by our extensive experimental results for different data analysis tasks over both real-life and synthetic datasets.

References

[1] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: A holistic solution to contingency table release. In *PODS*, 2007.

[2] D. L. Donoho and I. M. Johnstone. Ideal denoising in an orthonormal basis chosen from a library of bases. *Comptes Rendus Acad. Sci., Ser. I*, 319:1317–1322, 1994.

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

[4] S. Fienberg, A. Rinaldo, and X. Yang. Differential privacy and the risk-utility tradeoff for multi-dimensional contingency tables. In *PSD*, 2010.

[5] M. Hardt, K. Ligett, and F. McSherry. A simple and practical algorithm for differentially private data release. In *Technical Report, arXiv:1012.4763*, 2012.

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

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

[8] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevick, and T. Suel. Optimal histograms with quality guarantees. In *VLDB*, 1998.

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

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

[11] F. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *SIGMOD*, 2009.

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

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

[14] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *ICDE*, 2010.

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

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