Mean Estimation with User-Level Privacy for Spatio-Temporal IoT Datasets

Prajjwal Gupta, V. Arvind Rameshwar, Anshoo Tandon, and Novoneel Chakraborty

***Abstract*—This paper considers the problem of the private release of sample means of speed values from traffic datasets. Our key contribution is the development of user-level differentially private algorithms that incorporate carefully chosen parameter values to ensure low estimation errors on real-world datasets, while ensuring privacy. We test our algorithms on ITMS (Intelligent Traffic Management System) data from an Indian city, where the speeds of different buses are drawn in a potentially non-i.i.d. manner from an unknown distribution. We then apply our algorithms to a synthetic dataset, generated based on the ITMS data, having either a large number of users or a large number of samples per user. Here, we provide recommendations for the choices of parameters and algorithm subroutines that result in low estimation errors, while guaranteeing user-level privacy.**

I. Introduction

It is now well-understood that the release of even seemingly innocuous functions of a dataset that is not publicly available can result in the reconstruction of the identities of individuals (or users) in the dataset with alarming levels of accuracy (see, e.g., [1], [2] for attacks on traffic datasets). To alleviate concerns over such attacks, the framework of differential privacy (DP) was introduced in [3], which guarantees the privacy of users when each user contributes *at most one* sample. However, most real-world datasets, such as the traffic datasets, record multiple contributions from every user; naïvely applying standard DP techniques achieves poor estimation errors, owing to the addition of a large amount of noise to guarantee privacy. Recent work on "user-level privacy" [4] however demonstrates the effectiveness of some new algorithms that guarantee much improved estimation error, in comparison.

In this paper, we provide algorithms, which draw on [4], for ensuring user-level privacy in the context of releasing the sample means of speed records in traffic datasets. Our algorithms for estimating the sample means of the data crucially rely on carefully chosen procedures for clipping the number of samples contributed by each user, and for clipping each speed value so that it lies in a high-probability interval.

We first emperically evaluate the performance of such algorithms (via their estimation errors) on real-world speed values from ITMS (Intelligent Traffic Management System) traffic data, supplied by IoT devices deployed in an Indian city. Next, we generate a "large" synthetic dataset, using insights from the real-world ITMS data, with either a large number of users or a large number of samples per user. We demonstrate, via extensive experiments, the effectiveness or the relative poor performance of the different algorithms we employ, in each case. In addition, we provide theoretical justifications for the performance trends that we observe and recommendations for the choices of parameters and algorithm subroutines to be used on large real-world datasets.

P. G. contributed to this work during an internship at the IUDX Program Unit, SID, IISc, Bengaluru; he is currently with Northeastern University, email: gupta.praj@northeastern.edu. V. A. R., A. T., and N. C. are with the IUDX, emails: arvind.rameshwar@gmail.com, anshoo.tandon@gmail.com, novoneel.chakraborty@datakaveri.org.

II. Preliminaries

A. Notation

For a given $n \in \mathbb{N}$, the notation $[n]$ denotes the set $\{1, 2, \ldots, n\}$. Given a collection of real numbers $(x_1, \ldots, x_n)$, we define its median $\text{med}(x_1, \ldots, x_n)$ as any value x such that $|\{i \in [n] : x_i \geq x\}| = \lceil n/2 \rceil$. We use the notation $\text{Lap}(b)$ to refer to the zero-mean Laplace distribution with standard deviation $\sqrt{2}b$. We also use the notation $\mathcal{N}(\mu, \sigma^2)$ to denote the Gaussian distribution with mean μ and variance σ^2.

B. Problem Setup

The ITMS dataset that we consider contains records of the data provided by IoT sensors deployed in an Indian city containing, among other information, the license plate of the bus, the location at which the data was recorded, a timestamp, and the actual data value itself, which is the instantaneous speed of the bus. For the purpose of analysis, we divide the total area covered by bus routes into hexagon-shaped grids, with the aid of Uber's Hexagonal Hierarchical Spatial Indexing System (or H3) [5]. Furthermore, we quantize the timestamps present in the data records into 1 hour timeslots. In this work, we consider those data records that pertain to a single Hexagon And Timeslot (or HAT), and seek to privately release the average speeds of the buses in the chosen HAT. The algorithms discussed in this paper are applicable to general spatio-temporal IoT datasets for releasing a differentially private mean of the desired data values.

C. Problem Formulation

Fix a HAT of interest. Let L be the number of users (or distinct license plates) present in the HAT, and for every user $\ell \in [L]$, let the number of records contributed by the user be m_ℓ. We set $m^\star := \max_{\ell \in [L]} m_\ell$ and $m_\star := \min_{\ell \in [L]} m_\ell$. We assume that L and the collection $\{m_\ell : \ell \in [L]\}$ are known to the client. Now, let the collection $\left\{S_j^{(\ell)} : \ell \in [L], j \in [m_\ell]\right\}$ denote the speed values present in the records corresponding to the chosen HAT. We assume that each $S_j^{(\ell)} \in (0, U]$. For

the real-world ITMS dataset that we work with, the speed samples are drawn according to some unknown distribution P that is potentially non-i.i.d. (independent and identically distributed) across samples and users. However, when we generate synthetic speed samples, we draw each speed value $S_j^{(\ell)}$ i.i.d. according to some distribution P_s that is obtained by analyzing the statistics of the ITMS data.

Call the dataset consisting of the speed records of users present in the chosen HAT as $\mathcal{D} = \left\{\left(u_\ell, S_j^{(\ell)}\right) : \ell \in [L], j \in [m_\ell]\right\}$, where the collection $\{u_\ell : \ell \in [L]\}$ denotes the set of users.

The function that we are interested in computing is the sample average $f(\mathcal{D}) := \frac{1}{\sum_{\ell=1}^{L} m_\ell} \cdot \sum_{\ell=1}^{L} \sum_{j=1}^{m_\ell} S_j^{(\ell)}$.

D. User-Level Differential Privacy

Consider datasets $\mathcal{D}_1 = \left\{\left(u_\ell, x_j^{(\ell)}\right) : \ell \in [L], j \in [m_\ell]\right\}$ and $\mathcal{D}_2 = \left\{\left(u_\ell, \overline{x}_j^{(\ell)}\right) : \ell \in [L], j \in [m_\ell]\right\}$ consisting of the same users, with each user contributing the same number of (potentially different) data values $\{x_j^{(\ell)}\}$. Let D denote a universal set of such datasets. We say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are "user-level neighbours" if there exists $\ell_0 \in [L]$ such that $\left(x_1^{(\ell_0)}, \ldots, x_{m_{\ell_0}}^{(\ell_0)}\right) \neq \left(\overline{x}_1^{(\ell_0)}, \ldots, \overline{x}_{m_{\ell_0}}^{(\ell_0)}\right)$, with $\left(x_1^{(\ell)}, \ldots, x_{m_\ell}^{(\ell)}\right) = \left(\overline{x}_1^{(\ell)}, \ldots, \overline{x}_{m_\ell}^{(\ell)}\right)$, for all $\ell \neq \ell_0$.

Definition II.1. For a fixed $\epsilon > 0$, a mechanism $M : \mathsf{D} \to \mathbb{R}$ is said to be user-level ϵ-DP if for every pair of user-level neighbours $\mathcal{D}_1, \mathcal{D}_2$ and for every measurable subset $Y \subseteq \mathbb{R}$, we have that $\Pr[M(\mathcal{D}_1) \in Y] \leq e^\epsilon \Pr[M(\mathcal{D}_2) \in Y]$.

Definition II.2. Given a function $g : \mathsf{D} \to \mathbb{R}$, we define its user-level sensitivity Δ_g as $\Delta_g := \max_{\mathcal{D}_1, \mathcal{D}_2 \text{ u-l nbrs.}} |g(\mathcal{D}_1) - g(\mathcal{D}_2)|$, where the maximization is over datasets that are user-level neighbours.

For example, the user-level sensitivity of f is $\Delta_f = \frac{Um^\star}{\sum_\ell m_\ell}$. We use the terms "sensitivity" and "user-level sensitivity" interchangeably. The next result is well-known [3, Prop. 1].

Theorem II.1. *For any $g : \mathsf{D} \to \mathbb{R}$, the mechanism $M_g^{Lap} : \mathsf{D} \to \mathbb{R}$ defined by $M_g^{Lap}(\mathcal{D}_1) = g(\mathcal{D}_1) + Z$, where $Z \sim Lap(\Delta_g/\epsilon)$ and is independent of $\mathcal{D}$, is user-level ϵ-DP.*

Observe from Theorem II.1 that for a fixed $\epsilon > 0$, when $g = f$, the standard deviation of the noise added is proportional to the product $Um^\star$; in settings where either $m^\star$ or U is large, a prohibitively large amount of noise is added for privacy, thereby increasing the error in estimation.

III. Algorithms

We present 4 algorithms: Baseline, Array-Averaging, Levy, and Quantile, for private mean estimation with user-level privacy. Before we do so, we describe an approach that modifies the function f to be estimated, to reduce the noise added: we clip the number of samples contributed by any user ℓ to $\min\{m_\ell, m_{\text{UB}}\}$, where $m_\star \leq m_{\text{UB}} \leq m^\star$ is some fixed constant that depends on $\{m_\ell\}_{\ell \geq 1}$. We next describe two such "grouping" strategies, inspired by [6]: WrapAround and BestFit.

A. Strategies for Grouping Samples

Let m_{UB} be given. For ease of exposition, we first reindex the users so that $m_1 \geq m_2 \geq \ldots \geq m_L$. We then initialize L empty arrays $A_1, \ldots, A_L$, each of length m_{UB}. Let $w(A)$ denote the number of filled locations in an array A. Now, we process each user in turn, beginning with user 1 and populate his/her samples in the arrays, with a maximum of m_{UB} samples from any user being populated in the arrays. The WrapAround and BestFit strategies for populating arrays are explained below.

1) WrapAround*:* Let $\ell^\star$ denote the smallest value of $\ell \in [L]$ such that $m_\ell < m_{\text{UB}}$. For every $\ell < \ell^*$, we populate the array A_ℓ with the speed samples $\left(S_j^{(\ell)} : 1 \leq j \leq m_{\text{UB}}\right)$, in a contiguous manner. Next, for every user beginning with user $\ell^\star$, we populate his/her samples contiguously, starting from the first empty location, breaking a user's samples into two arrays if needed. Let K denote the index of the last fully filled array, with the arrays $A_{K+1}, \ldots, A_L$ being deleted. Note that $K = \left\lfloor \frac{\sum_{\ell=1}^{L} \min\{m_\ell, m_{\text{UB}}\}}{m_{\text{UB}}} \right\rfloor$. Observe that in this strategy, a user can "influence" two arrays, in that his/her samples can lie in two arrays A_i and A_{i+1}, for some $1 \leq i \leq K-1$.

2) BestFit*:* This algorithm is a version of a popular online algorithm for the bin-packing problem. Similar to the description of WrapAround, let $\ell^\star$ denote the smallest value of $\ell \in [L]$ such that $m_\ell < m_{\text{UB}}$. For every $\ell < \ell^\star$, we populate the array A_ℓ with the speed samples $\left(S_j^{(\ell)} : 1 \leq j \leq m_{\text{UB}}\right)$, in a contiguous manner. For each user $\ell \geq \ell^\star$, we find the least-indexed array A among $A_{\ell^\star}, \ldots, A_L$ that can accommodate m_ℓ samples and is filled the most. The m_ℓ samples from the user ℓ are then placed in array A contiguously and the process is iterated over the other users. Note that the number of non-empty arrays that are returned by the BestFit algorithm $\overline{K}$, is at least K. Furthermore, unlike in the WrapAround strategy, any user influences at most one array.

We shall now describe our main algorithms.

B. Baseline

The Baseline algorithm computes $M_{\text{Baseline}}(\mathcal{D}) = f(\mathcal{D}) + \text{Lap}(\Delta_f/\epsilon)$, where $\Delta_f = \frac{Um^\star}{\sum_\ell m_\ell}$ is the sensitivity of the function f. As mentioned earlier, this algorithm suffers from the drawback that a large amount of noise needs to be added for privacy when either U or $m^\star$ is large.

C. Array-Averaging

In this algorithm, we attempt to reduce the amount of noise added when $m^\star$ is large, by clipping the number of samples contributed by any given user to some $m_{\text{UB}} \in [m_\star, m^\star]$. We use one of WrapAround or BestFit for grouping. In particular, given the dataset $\mathcal{D}$, we set $f_{\text{arr, wrap}}(\mathcal{D}) := \frac{1}{K} \cdot \sum_{i=1}^{K} \overline{A_i}$, where A_i are the arrays obtained using the WrapAround strategy, with $\overline{A_i} := \frac{1}{w(A_i)} \sum_{j=1}^{w(A_i)} A_i(j)$ being the mean of the samples contributed by array $i \in [K]$. We also set $f_{\text{arr, best}}(\mathcal{D}) := \frac{1}{\overline{K}} \cdot \sum_{i=1}^{\overline{K}} \overline{A_i}$, where A_i are the arrays obtained using the BestFit strategy.

Since using the WRAPAROUND strategy can result in one user influencing two arrays, the sensitivity is given by $\Delta_{f_{\text{arr, wrap}}} = \frac{2U}{K}$. However, using the BESTFIT strategy, the sensitivity is $\Delta_{f_{\text{arr, best}}} = \frac{U}{\overline{K}}$.

The ARRAY-AVERAGING algorithm with WRAPAROUND grouping computes $M_{\text{ArrayAvg, wrap}}(\mathcal{D}) = f_{\text{arr, wrap}}(\mathcal{D}) + \text{Lap}(\Delta_{f_{\text{arr, wrap}}}/\epsilon)$, and with BESTFIT grouping computes $M_{\text{ArrayAvg, best}}(\mathcal{D}) = f_{\text{arr, best}}(\mathcal{D}) + \text{Lap}(\Delta_{f_{\text{arr, best}}}/\epsilon)$. Clearly, both these algorithms are ϵ-DP, from Theorem II.1.

Observe that since $\overline{K} \geq K$, we have that $\Delta_{f_{\text{arr, best}}} \leq \frac{U}{K} < \Delta_{f_{\text{arr, wrap}}}$. This suggests that using the BESTFIT grouping strategy yields lower estimation error, since the corresponding standard deviation of the noise added is lower by a multiplicative factor of 2 compared to the WRAPAROUND strategy. In what follows, we therefore focus our attention on the BESTFIT strategy.

We now define $\tilde{\Delta}_{f_{\text{arr}}} := \frac{Um_{\text{UB}}}{\sum_{\ell=1}^{L} \min\{m_\ell, m_{\text{UB}}\}}$ as a proxy for the upper bound U/K on $\Delta_{f_{\text{arr, best}}}$. If $\tilde{\Delta}_{f_{\text{arr}}}$ is small, then so are U/K and $\Delta_{f_{\text{arr, best}}}$.

The following lemma then holds:

Lemma III.1. *For any $m_{UB} \leq m^\star$, we have that $\Delta_f \geq \tilde{\Delta}_{f_{arr}}$.*

Proof. We need only prove that $\frac{m^\star}{\sum_\ell m_\ell} \geq \frac{m_{\text{UB}}}{\sum_{\ell=1}^{L} \min\{m_\ell, m_{\text{UB}}\}}$, or equivalently, that $\frac{m^\star}{m_{\text{UB}}} \geq \frac{\sum_\ell m_\ell}{\sum_{\ell=1}^{L} \min\{m_\ell, m_{\text{UB}}\}}$. To see this, let $B := \{\ell \in [L] : m_\ell < m_{\text{UB}}\}$ and $B^c := [L] \setminus B$. Then,

$$\frac{\sum_\ell m_\ell}{\sum_{\ell=1}^{L} \min\{m_\ell, m_{\text{UB}}\}} = \frac{\sum_{\ell \in B} m_\ell + \sum_{\ell \in B^c} m_\ell}{\sum_{\ell \in B} m_\ell + \sum_{\ell \in B^c} m_{\text{UB}}} \leq \frac{\sum_{\ell \in B^c} m_\ell}{|B^c| \cdot m_{\text{UB}}} \leq \frac{m^\star}{m_{\text{UB}}},$$

where the first inequality holds by the definition of B^c and the second inequality holds since $m_\ell \leq m^\star$, for all $\ell \in [L]$. □

Next, we shall embark on choosing a "good" m_{UB}, which provides a large "gain" $\frac{\Delta_f}{\tilde{\Delta}_{f_{\text{arr}}}}$. To this end, call $\alpha := \frac{m^\star}{m_{\text{UB}}}$; in our setting, we have $\alpha \geq 1$. For fixed $\{m_\ell\}_{\ell \geq 1}$,

$$\frac{\Delta_f}{\tilde{\Delta}_{f_{\text{arr}}}} = \frac{\alpha}{\sum_\ell m_\ell} \cdot \sum_\ell \min\{m_\ell, m_{\text{UB}}\} \leq \frac{\alpha}{\sum_\ell m_\ell} \cdot \min\left\{\sum_\ell m_\ell, m^\star L/\alpha\right\} = \min\left\{\alpha, \frac{m^\star L}{\sum_\ell m_\ell}\right\}.$$

In the above, the equality $\frac{\Delta_f}{\tilde{\Delta}_{f_{\text{arr}}}} = \alpha$ holds only if $m_{\text{UB}} \geq m_\ell$, for all $\ell \in [L]$, or equivalently, $m_{\text{UB}} = m^\star$. Note that in this case $\alpha = \frac{\Delta_f}{\tilde{\Delta}_{f_{\text{arr}}}}$ in fact equals 1. On the other hand, the equality $\frac{\Delta_f}{\tilde{\Delta}_{f_{\text{arr}}}} = \frac{m^\star L}{\sum_\ell m_\ell} \geq 1$ holds only if $m_{\text{UB}} \leq m_\ell$, for all $\ell \in [L]$, or equivalently, $m_{\text{UB}} = m_\star$. We designate $\mathsf{OPT} := \frac{m^\star L}{\sum_\ell m_\ell}$, and note that $\Delta_f = \mathsf{OPT} \cdot \tilde{\Delta}_{f_{\text{arr}}}$ when $m_{\text{UB}} = m_\star$.

While the choice $m_{\text{UB}} = m_\star$ results in a high gain, it could potentially cause poor accuracy in estimation of the true value $f(\mathcal{D})$, since each array contains only very few samples. The next lemma shows that choosing the more favourable $m_{\text{UB}} = \text{med}(m_1, \ldots, m_L)$ results in a gain that is only at most a factor of 2 lower than OPT.

Lemma III.2. *The choice $m_{UB} = med(m_1, \ldots, m_L)$ results in $\frac{\Delta_f}{\tilde{\Delta}_{farr}} \geq \frac{\mathsf{OPT}}{2}$.*

Proof. For this choice of m_{UB}, we have that, setting $\alpha = \frac{m^\star}{m_{\text{UB}}}$ and $B := \{\ell \in [L] : m_\ell < m_{\text{UB}}\}$,

$$\frac{\Delta_f}{\tilde{\Delta}_{f_{\text{arr}}}} = \frac{\alpha}{\sum_\ell m_\ell} \cdot \sum_\ell \min\{m_\ell, m_{\text{UB}}\} = \frac{\alpha}{\sum_\ell m_\ell} \cdot \left[\sum_{\ell \in B} m_\ell + \frac{m^\star \lceil L/2 \rceil}{\alpha}\right] \geq \frac{\mathsf{OPT}}{2}.$$

□

In our experiments in Section IV, we employ $m_{\text{UB}} = \text{med}(m_1, \ldots, m_L)$ for ARRAY-AVERAGING under the BESTFIT grouping strategy.

D. LEVY

In this algorithm and the next, we attempt to reduce Δ_f further by simultaneously clipping the number of samples per user, and the range of speed values. The algorithm that we present in this section puts together ARRAY-AVERAGING and Algorithm 1 in [4]. However, unlike in the previous section, we make use of a value of m_{UB} that is tailored, via heuristic calculations, to the sensitivity of the function associated with this algorithm. We now present a qualitative description of the algorithm; the exact choice of the parameters is based on certain heuristics that will be described later.

Let $A_1, \ldots, A_{\overline{K}}$ be the arrays obtained using the BESTFIT grouping strategy with a certain value of m_{UB} to be specified later (recall the discussion in Section III-A on the advantages of BESTFIT over WRAPAROUND). Next, we clip the range of speed values. The LEVY algorithm first privately estimates an interval $[a, b]$ where the speed values lie with high probability, with privacy loss set to $\epsilon/2$.

We then define the function $f_{\text{Levy}}(\mathcal{D}) := \frac{1}{\overline{K}} \cdot \sum_{i=1}^{\overline{K}} \Pi_{[a,b]}(\overline{A_i})$, where $\Pi_{[a,b]}$ denotes the projection operator into the interval $[a, b]$, with $\Pi_{[a,b]}(x) = \min\{b, \max\{a, x\}\}$, for any $x \in \mathbb{R}$. As before, $\overline{A_i} = \frac{1}{w(A_i)} \sum_{j=1}^{w(A_i)} A_i(j)$. Note that now the user-level sensitivity is given by $\Delta_{f_{\text{Levy}}} = \frac{b-a}{\overline{K}}$.

The LEVY algorithm then computes $M_{\text{Levy}}(\mathcal{D}) = f_{\text{Levy}}(\mathcal{D}) + \text{Lap}(2\Delta_{f_{\text{Levy}}}/\epsilon)$. Note that in the above expression, the privacy loss is assumed to be $\epsilon/2$. Overall, the privacy loss for both private interval estimation and for private mean estimation is ϵ, following the basic composition theorem [7, Corollary 3.15]. Hence, the algorithm LEVY is ϵ-DP.

Observe that when $b - a < U/2$, the standard deviation of the noise added in this case is less than that added using ARRAY-AVERAGING with BESTFIT grouping. We now explain the heuristics we employ to select the parameters a, b in the LEVY algorithm and m_{UB}.

1) Private Interval Estimation: The subroutine we use in this algorithm to privately compute the "high-probability" interval $[a, b]$ is borrowed from Algorithm 6 in [4], which crucially relies on the following concentration property of the data values provided to the algorithm:

Definition III.1. A random sequence X^n supported on $[0, M]$ is (τ, γ)-concentrated (τ is called the "concentration radius") if there exists $x_0 \in [0, M]$ such that with probability at least $1 - \gamma$, $\max_{i \in [n]} |X_i - x_0| \leq \tau$.

In what follows, we set $\gamma = 0.2$. Although the samples in each array are drawn in a potentially non-i.i.d. fashion from an unknown distribution, we simply rely on heuristics that assume that the data samples are i.i.d. and that each array is fully filled, with $\overline{K} = K$.

By an application of Hoeffding's inequality (see, e.g., [8, Theorem 2.2.6]), we have that each array mean $\overline{A}_i$, $i \in [K]$, is $\frac{U^2}{4m_{\mathrm{UB}}}$-subGaussian (see, e.g., [9, Theorem 2.1.1]). Hence, if the speed samples were i.i.d., we obtain (see, e.g., [9, Theorem 2.2.1]) that the sequence $(\overline{A}_1, \ldots, \overline{A}_K)$ is in fact (τ, γ)-concentrated about the expected value, where $\tau = U \cdot \sqrt{\frac{\log(2K/\gamma)}{2m_{\mathrm{UB}}}}$. We use this value of τ to compute the "high-probability" interval $[a, b]$, with privacy loss $\epsilon/2$. More precisely, we divide the interval $(0, U]$ into U/τ disjoint bins, each of width τ, and use these bins to compute an estimate $\hat{\mu}$ of the median of the data samples as in Algorithm 6 of [4]. We then set $a = \hat{\mu} - \frac{3\tau}{2}$ and $b = \hat{\mu} + \frac{3\tau}{2}$.

2) Choosing m_{UB}: The subroutine we use to choose the length of the arrays m_{UB} is tailored to the sensitivity of the function f_{Levy}. For a fixed $m = m_{\mathrm{UB}}$, note that $\Delta_{f_{\mathrm{Levy}}}(m) = \frac{3\tau}{K} = \frac{3U}{K}\sqrt{\frac{\log(2K/\gamma)}{2m}}$. To reduce the sensitivity, our heuristic aims to maximize the $K\sqrt{m}$ term in $\Delta_{f_{\mathrm{Levy}}}(m)$, and sets

$$m_{\mathrm{UB}} = \mathrm{argmax}_{m_\star \leq m \leq m^\star} \frac{\sum_{\ell=1}^{L} \min\{m_\ell, m\}}{\sqrt{m}}, \qquad (1)$$

and numerically solves this optimization problem.

E. QUANTILE

The QUANTILE algorithm sets m_{UB} as in (1) and uses the BESTFIT grouping strategy to create arrays containing the speed samples. Next, the algorithm privately estimates a "high-probability" interval $[a', b']$ (different from the one used in LEVY) with privacy loss $\epsilon/2$. We then define $f_{\mathrm{Quantile}}(\mathcal{D}) := \frac{1}{K} \cdot \sum_{i=1}^{K} \Pi_{[a',b']}(\overline{A}_i)$, with the sensitivity $\Delta_{f_{\mathrm{Quantile}}} = \frac{b'-a'}{K}$, The QUANTILE algorithm then computes $M_{\mathrm{Quantile}}(\mathcal{D}) = f_{\mathrm{Quantile}}(\mathcal{D}) + \mathrm{Lap}(2\Delta_{f_{\mathrm{Quantile}}}/\epsilon)$, where the privacy loss for mean estimation is set to $\epsilon/2$. We describe two subroutines that we use to privately estimate $[a', b']$.

1) FIXEDQUANTILE*:* This subroutine privately estimates the $\left(\frac{1}{10}, \frac{9}{10}\right)$-interquantile interval of the array means $\overline{A}_1, \ldots, \overline{A}_{\overline{K}}$, using Algorithm 2 in [10]. Note that this algorithm computes estimates of a' and b' separately, and we set the privacy loss of each of these computations to be $\epsilon/4$, so that the overall privacy loss for quantile estimation is $\epsilon/2$.

2) ε-DEPENDENTQUANTILE*:* This subroutine privately estimates (with privacy loss $\epsilon/2$) the interval $[a', b']$, which is now chosen to minimize the sum of the absolute estimation errors due to privacy and due to clipping the speed values, following the work in [11]. In particular, an argument similar to that in Section 3 in [11] advocates that in order to minimize this sum of absolute estimation errors, we need to set b to be the $\left\lceil \frac{2}{\epsilon} \right\rceil^{\text{th}}$-largest value among $(\overline{A}_1, \ldots, \overline{A}_{\overline{K}})$, and by symmetry, we need to set a to be the $\left\lfloor \frac{2}{\epsilon} \right\rfloor^{\text{th}}$-smallest value among $(\overline{A}_1, \ldots, \overline{A}_{\overline{K}})$. We then privately estimate the $\left(\frac{1}{K} \cdot \left\lceil \frac{2}{\epsilon} \right\rceil, 1 - \frac{1}{K} \cdot \left\lceil \frac{2}{\epsilon} \right\rceil\right)$-interquantile interval using Algorithm 2 in [10]. Again, we set the privacy loss of computing a' and b' to be individually $\epsilon/4$, so that the overall privacy loss for private quantile estimation is $\epsilon/2$.

IV. RESULTS

A. Setup

We test and evaluate our algorithms on two types of datasets: 1) a real-world ITMS dataset containing non-i.i.d. speed values contributed by bus drivers over a span of 3 days and 2) a synthetic dataset containing i.i.d. samples drawn using insights from the ITMS data. We ran each private mean estimation algorithm described earlier 10^4 times; the plots showing the performance of our algorithms are shown in this section. We test the accuracy of our algorithms, via the error in estimation, for privacy loss ϵ ranging from 0.5 to 2.

We use the mean absolute error (or MAE) metric, $E_{\mathrm{MAE}}(C) = \frac{\sum_{i=0}^{10^4} |M^{(i)}(C) - \mu(C)|}{10^4}$, to evaluate the performance of the algorithms. Here, for $i \in [10^4]$, $M^{(i)}(C)$ is the result of running the algorithm M (which is one of ARRAY-AVERAGING, LEVY, and QUANTILE) on the dataset C at iteration i, and $\mu(C)$ is the true sample mean. Since, for $Z \sim \mathrm{Lap}(b)$, we have $\mathbb{E}[|Z|] = b$, we simply use $E_{\mathrm{MAE}}(C) = \Delta_f/\varepsilon$ for BASELINE.

B. Experimental Results for ITMS Data

For our ITMS dataset $\mathcal{D}$, we have $m^* = 417$, $\sum_\ell m_\ell = 17166$, and $U = 65$ km/hr. The true sample mean of speed records $\mu(\mathcal{D})$ was 20.66769. We also mention that we pick $\mathrm{med}(m_1, \ldots, m_L) = 46$. Also, for m_{UB} as in (1), the number of arrays K under WRAPAROUND grouping was 160 and the number of arrays $\overline{K}$ under BESTFIT grouping was 164.

Due to the paucity of space, we have omitted the numerical results showing the superior performance of BESTFIT as compared to WRAPAROUND, for all algorithms described. We hence use BESTFIT, in what follows, as the grouping strategy for all our numerical comparisons. Figure 1, for the non-i.i.d. ITMS dataset, shows that clipping results in much better performance than the naïve BASELINE approach.

C. Experimental Results for Synthetic Data

Next, we artificially generate data samples, by drawing insights from the ITMS dataset. This dataset allows us to study the performance of the algorithms in Section III when the number of users, denoted by $\hat{L}$, or the number of samples per user, denoted by $\{\hat{m}_\ell\}_{\ell \in [\hat{L}]}$, is large. We generate i.i.d. speed samples $\left\{\hat{S}_j^{(\ell)} : \ell \in [\hat{L}], j \in [\hat{m}_\ell]\right\}$, where for any ℓ, j, we have $\hat{S}_j^{(\ell)} \sim \Pi_{[0,U]}(X)$, where $X \sim \mathcal{N}(\mu(\mathcal{D}), \sigma^2(\mathcal{D}))$. Here, $\mu(\mathcal{D})$ is the true sample mean of the ITMS dataset and $\sigma^2(\mathcal{D})$ is the true sample variance of the ITMS dataset, defined as $\sigma^2(\mathcal{D}) = \frac{1}{\sum_{\ell=1}^{L} m_\ell} \cdot \sum_{\ell=1}^{L} \sum_{j=1}^{m_\ell} \left(S_j^{(\ell)} - \mu(\mathcal{D})\right)^2$. We consider two settings:

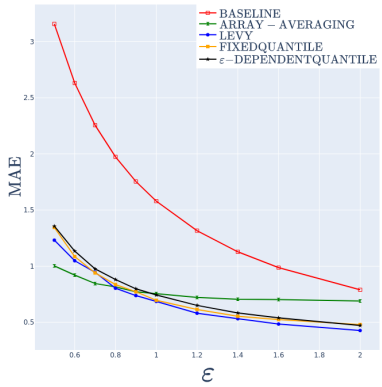
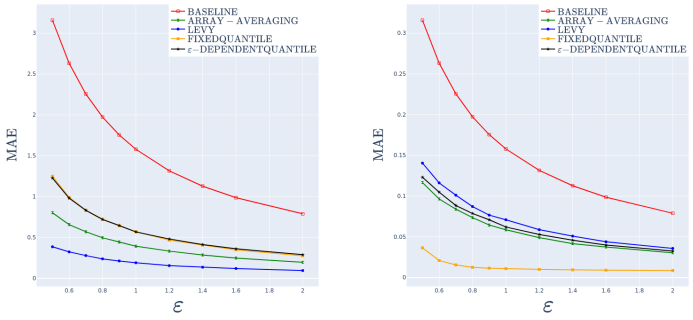


Fig. 1: Plots comparing the performance of algorithms on real-world ITMS data



(a) (b)

Fig. 2: (a) Plots comparing the performance of algorithms under **sample scaling** (b) Plots comparing the performance of algorithms under **user scaling**

1) **Sample scaling**: Here, $\hat{L} = L$ and $\hat{m}_\ell = 10 \cdot m_\ell$, for all $\ell \in [\hat{L}]$, and $\hat{m}^\star := \max_{\ell \in [\hat{L}]} \hat{m}_\ell = 10 \cdot m^\star$.
2) **User scaling**: Here, $\hat{L} = 10 \cdot L$ and for each $i \in [10]$, we let $\hat{m}_{10\cdot(\ell-1)+i} = m_\ell$, for all $\ell \in [L]$, with $\hat{m}^\star = m^\star$.

Figure 2a for **sample scaling** shows that BASELINE performs worse than the other algorithms, as expected. The justification for the relative superior performance of LEVY in the sample scaling setting is as follows. First, observe that in LEVY, we have that $\hat{m}_{\text{UB}} = 10 \cdot m_{\text{UB}}$ solves (1), where m_{UB} was the length of the arrays used in the absence of sample scaling. Furthermore, the number of arrays $\hat{\overline{K}}$ obtained using BESTFIT grouping equals $\overline{K}$. Hence, $\Delta_{f_{\text{Levy}}}(\hat{m}_{\text{UB}}) = \frac{3U}{\overline{K}}\sqrt{\frac{\log(2\overline{K}/\gamma)}{20 m_{\text{UB}}}}$, which is a factor $1/\sqrt{10}$ smaller than the user-level sensitivity without scaling. In contrast, the sensitivities for BASELINE and ARRAY-AVERAGING remain unchanged from the corresponding values without sample scaling. Hence, in general, when the dataset contains a large number of samples per user, we recommend using LEVY.

Figure 2b for **user scaling** shows that the FIXEDQUANTILE subroutine outperforms all the other algorithms. We mention here that we use m_{UB} as in (1) for ARRAY-AVERAGING as well, to maintain uniformity. To justify the performance trends, note that for LEVY, τ (and hence $\Delta_{f_{\text{Levy}}}$) increases (logarithmically) with the number of users. This implies that when the number of users are sufficiently large, the computed interval $[a, b]$ will be equal to the entire interval $(0, U]$. Next, for ε-DEPENDENTQUANTILE, note that since $\epsilon \geq 0.5$, we have that the interval $[a', b']$ excludes $\lfloor 4/\varepsilon \rfloor \leq 8$ outlier speed samples, suggesting that for typical datasets with a large number of users, $b' - a' \approx U$. In comparison, the FIXEDQUANTILE subroutine eliminates a much larger number of data samples in the computation of $[a', b']$, implying that the amount of noise added in this case is low. Hence, in general, for datasets with a large number of users, we recommend using FIXEDQUANTILE.

V. CONCLUSION

We proposed algorithms for the private release of sample means of real-world datasets, with particular focus on traffic datasets that contain speed data samples. We analyzed the performance of the different algorithms proposed, via extensive experiments, provided theoretical justification for the choices of subroutines used, and recommended subroutines for large datasets. An interesting line of future research is to extend the results in this work to the private release of sample means from several grids in the city simultaneously.

REFERENCES

[1] C. Whong. (2014) Foiling nyc's taxi trip data. [Online]. Available: https://chriswhong.com/open-data/foil_nyc_taxi/

[2] V. Pandurangan. (2014) On taxis and rainbow tables: Lessons for researchers and governments from nyc's improperly anonymized taxi logs. [Online]. Available: https://blogs.lse.ac.uk/impactofsocialsciences/2014/07/16/nyc-improperly-anonymized-taxi-logs-pandurangan/

[3] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," *Theory of Cryptography*, p. 265–284, 2006.

[4] D. A. N. Levy, Z. Sun, K. Amin, S. Kale, A. Kulesza, M. Mohri, and A. T. Suresh, "Learning with user-level privacy," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021. [Online]. Available: https://openreview.net/forum?id=G1jmxFOtY_

[5] (Uber Technologies Inc.) H3: Hexagonal hierarchical geospatial indexing system. [Online]. Available: https://h3geo.org/

[6] A. J. George, L. Ramesh, A. Vikram Singh, and H. Tyagi, "Continual Mean Estimation Under User-Level Privacy," *arXiv e-prints*, p. arXiv:2212.09980, Dec. 2022.

[7] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014. [Online]. Available: http://dx.doi.org/10.1561/0400000042

[8] R. Vershynin, *High-Dimensional Probability: An Introduction with Applications in Data Science*, ser. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2018.

[9] E. Pauwels, "Statistics and optimization in high dimensions," Lecture notes. [Online]. Available: https://www.math.univ-toulouse.fr/~epauwels/M2RI/session1.pdf

[10] A. D. Smith, "Privacy-preserving statistical estimation with optimal convergence rates," in *STOC '11: Proceedings of the forty-third annual ACM symposium on Theory of computing*, 2011. [Online]. Available: https://dl.acm.org/doi/10.1145/1993636.1993743

[11] K. Amin, A. Kulesza, A. Munoz, and S. Vassilvtiskii, "Bounding user contributions: A bias-variance trade-off in differential privacy," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 263–271. [Online]. Available: https://proceedings.mlr.press/v97/amin19a.html