Private Analysis of Infinite Data Streams via Retroactive Grouping

Rui Chen
Samsung Research America
Mountain View, CA, USA
rui.chen1@samsung.com

Yilin Shen
Samsung Research America
Mountain View, CA, USA
yilin.shen@samsung.com

Hongxia Jin
Samsung Research America
Mountain View, CA, USA
hongxia@acm.org

ABSTRACT

With the rapid advances in hardware technology, data streams are being generated daily in large volumes, enabling a wide range of real-time analytical tasks. Yet data streams from many sources are inherently sensitive, and thus providing continuous privacy protection in data streams has been a growing demand. In this paper, we consider the problem of private analysis of infinite data streams under differential privacy. We propose a novel data stream sanitization framework that periodically releases histograms summarizing the event distributions over sliding windows to support diverse data analysis tasks. Our framework consists of two modules, a sampling-based change monitoring module and a continuous histogram publication module. The monitoring module features an adaptive Bernoulli sampling process to accurately track the evolution of a data stream. We for the first time conduct error analysis of sampling under differential privacy, which allows to select the best sampling rate. The publication module features three different publishing strategies, including a novel technique called retroactive grouping to enjoy reduced noise. We provide theoretical analysis of the utility, privacy and complexity of our framework. Extensive experiments over real datasets demonstrate that our solution substantially outperforms the state-of-the-art competitors.

Categories and Subject Descriptors

H.2.7 [**DATABASE ADMINISTRATION**]: Security, integrity & protection

Keywords

Differential privacy, data stream, sampling-based monitoring, retroactive grouping

1. INTRODUCTION

With the rapid advances in hardware technology, there have been numerous applications that create and/or consume data streams in real time. Examples include public health surveillance, IoT device data, mobile data, GPS data, call detail records (CDRs), etc. Sharing such streams enables a wide spectrum of data analysis tasks in a real-time manner. However, streaming data from many sources are inherently sensitive, making privacy protection an increasingly important demand [17].

One notable privacy risk of sanitizing streaming data is that individually sanitized data could jointly compromise privacy. Fortunately, this risk can be gracefully addressed by the differential privacy model [8], which provides composable privacy guarantees for continuous data publishing over time. In the context of data streams, there are two instantiations of differential privacy providing different extents of privacy protection: *event-level* differential privacy and *user-level* differential privacy [9]. Event-level differential privacy hides any single event of any user from attackers, while user-level differential privacy hides all events of any user. In this paper, we follow the conventional use of event-level differential privacy [3, 5, 6, 9] mainly due to its ability to support private analysis of *infinite* data streams with reasonable accuracy. Infinite monitoring/analysis is a practical requirement in many applications. Meanwhile, we stress that event-level differential privacy still provides strong and meaningful privacy protection in that it provides *any* individual *plausible deniability* for *any* event in the published results.

While there have been a few existing works [3–6, 9–11, 13] on applying differential privacy to data streams, our problem setting is different from theirs and is oriented toward practical applications. We consider a more realistic data stream model that is widely used in real-world applications: periodically releasing synopsis structures (e.g., histograms) of sliding windows over practical infinite data streams (not simplified 0/1 strings as in some previous papers) [2]. Due to the consideration of practical usefulness, we advocate the *data-dependent* paradigm, which has been shown to be able to obtain significantly better utility in many cases [14]. The general idea of the data-dependent paradigm is to adaptively make use of the properties of the underlying data to guide private operations that lead to lower error.

More specifically, our solution is inspired by the power of grouping- or clustering-based approaches on *static* histogram publication [1, 18, 19]. These approaches group or cluster the bins of a histogram with similar counts to enjoy Laplace noise reduction proportional to the sizes of the groups or clusters. In the context of data streams, it is natural to attempt a similar idea to the time dimension: group time units with similar trends to improve utility. However,

CIKM'15, October 19-23, 2015, Melbourne, VIC, Australia.
© 2015 ACM. ISBN 978-1-4503-3794-6/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2806416.2806454.

applying this idea to a data stream faces several non-trivial technical challenges: (1) The data elements (or events) arrive in real time. At the time of publication, we do not have knowledge about future trends. (2) It is required to release the histogram of the current sliding window at each timestamp. We cannot postpone publication to collect a batch of histograms to analyze their similarity. (3) Grouping with previously published histograms might increase the sensitivity and thus the magnitude of noise. In view of these challenges, we design a data stream sanitization framework equipped with novel sampling and publishing techniques to improve utility.

Contributions. In this paper, we consider a practical setting for continuously analyzing infinite data streams with the differential privacy guarantee. In addressing the problem, we first propose two first-cut solutions, accompanied with formal utility analysis, and then propose a novel data stream sanitization framework that is composed of two major modules: a sampling-based change monitoring module and a continuous histogram publication module.

The novelty of the monitoring module lies in an adaptive Bernoulli sampling process that accurately detects changes of the data stream even with a small privacy budget. While allowing to reduce Laplace error, sampling itself introduces a new source of sampling error, which is neglected in previous research. However, it needs to be carefully studied to properly adjust the sampling rate so as to improve accuracy. We for the first time quantify the sampling error and identify the best sampling rate as an optimization problem. Such analysis is useful for many sampling-based algorithms under differential privacy.

The publication module features three different publishing strategies, including a novel technique called *retroactive grouping*. It sophisticatedly selects time units to passively publish (i.e., approximate their statistics without explicitly using the Laplace mechanism) and then retroactively groups these units with later time units to enjoy reduced noise. We formally prove that retroactive grouping satisfies differential privacy. We theoretically analyze the accuracy of each released histogram and the space and time complexity of our solution to demonstrate its suitability for processing large-scale real-time data streams. We stress that while we focus on histograms in this paper, our general framework can easily accommodate other synopsis structures.

We have conducted an extensive experimental study over real datasets. The results confirm that our solution achieves substantially better utility than different competitors.

2. RELATED WORK

Applying differential privacy to data streams is a very recent topic. The first research papers [6, 9] consider computing the running sum of 1's (i.e., the number of 1's seen so far) over a simplified stream of 0's and 1's whose length is T. They independently present similar results. The proposed schemes encode the partial sums in terms of a full binary tree, where each node stores a partial sum plus noise with scale logarithmic in T, and answer the running sum by using the minimum number of nodes in the binary tree. Bolot et al. [3] extends the previous papers by considering decayed sum. A dyadic tree structure is maintained in a non-uniform manner. Our solution is orthogonal to these methods, but they can be combined to enhance each other.

Fan et al. [10] consider user-level differential privacy on *finite* streams. They present the FAST framework whose key idea is to adaptively sample the timestamps and take different publishing strategies for sampled and non-sampled timestamps in order to increase the privacy budget allocated to sampled timestamps. For sampled timestamps FAST releases their perturbed values by using the Laplace mechanism, while for non-sampled timestamps FAST releases their predicted counts instead. FAST is not directly beneficial to our problem setting because not sampling a timestamp does not increase other timestamps' privacy budgets. Also, the prediction accuracy could be low sometimes.

Kellaris et al. [13] propose a variant of event-level differential privacy called w-event differential privacy. It hides the presence/absence of any consecutive event sequence of size w. They introduce two privacy budget allocation schemes, Budget Distribution (BD) and Budget Absorption (BA), to achieve w-event differential privacy over infinite streams. Their contributions are orthogonal to ours. We aim at developing techniques to improve accuracy, which could be seamlessly incorporated into the w-event differential privacy framework.

Friedman et al. [11] study the problem of privately monitoring arbitrary *threshold* functions over *distributed* finite streams. This is achieved by the concept of Safe Zones, within which local vectors are guaranteed to satisfy the global privacy condition. Compared to theirs, our problem is more challenging in that we need to publish data, not merely signaling the exceeding of a given threshold. Similar to the distributed setting in [11], Chan et al. [5] study the problem of continual monitoring of heavy hitters from distributed streams by an untrusted aggregator.

A different problem setting on data streams is studied in [4]. Cao et al. consider to answer a given set of correlated sliding window aggregate queries. They propose to first sample a subset of representative queries, which will be answered by adding Laplace noise, and then estimate the answers to the remaining queries from the answers to the representatives.

As mentioned earlier, our work is motivated by studies on *static* histogram publication [1,18,19]. Xu et al. [18] propose to differentially privately group a histogram's bins based on the traditional histogram construction technique. One disadvantage of their solution is the requirement of knowing the number of groups in advance. Acs et al. [1] circumvent this drawback by proposing a bisection-based grouping strategy that automatically finds a good number of groups. Zhang et al. [19] argue that global clustering can achieve better utility than the local grouping schemes in [1, 18] and consequently propose a clustering framework that carefully balances the approximation error and the Laplace error. Unfortunately, such a global clustering scheme does not apply to data streams due to the challenges mentioned in the previous section.

3. PROBLEM FORMULATION

In this section, we first introduce the data stream model and the privacy notion in use and then give our problem statement.

3.1 Data Stream Model

In this paper, we consider an *infinite* data stream $\mathcal{S}$ consisting of an infinite number of events that arrive in real

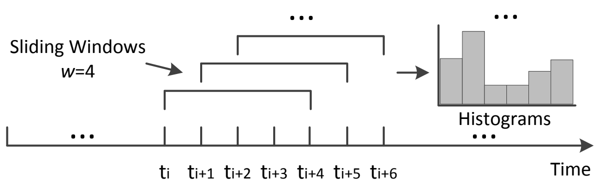


Figure 1: A sample data stream

time. Each event belongs to a user and is associated with an attribute value. The data publisher (e.g., the online recommender system in Example 1) continuously monitors the stream and publishes histograms summarizing the event distributions in the latest sliding windows at discrete timestamps. We consider publishing histograms as they have wide applications in a broad spectrum of problems [2]. Each bin B_i in a histogram $\mathbf{H}$ corresponds to an attribute value, and its count H_i is the number of events falling into B_i within the sliding window. We denote the total number of bins in $\mathbf{H}$ by $|\mathbf{H}|$. We call the time duration between two consecutive timestamps a *time unit*. The size of a sliding window is the number of time units it covers, which is denoted by ω. We denote the segment of $\mathcal{S}$ between timestamps t_i and t_j by $\mathcal{S}_{t_i}^{t_j}$, where $i < j$. Such a data stream model corresponds to many real-life applications, as illustrated below.

EXAMPLE 1. *For an online recommender system (e.g., Yelp), it is a business routine to continuously monitor and analyze users' events (e.g., users' visits to different types of restaurants). Periodically releasing histograms summarizing user events in the latest sliding windows (e.g., the distribution of restaurant types being visited in the past two weeks) is beneficial in many aspects, most importantly as a building block for recommendations. Such a scenario is illustrated in Figure 1, where the recommender system publishes the histograms of visits to different types of restaurants within the past 4 time units at every discrete timestamp. Each bin in a histogram corresponds to a restaurant type and has a count equal to the number of visits within the sliding window.*

3.2 Privacy Model

In the context of data streams, differential privacy rests on the notion of indistinguishability between two *neighboring* streams. In this paper, we follow the previous studies [3, 5, 6, 9] to define two data streams $\mathcal{S}$ and $\mathcal{S}'$ as neighbors if $\mathcal{S}$ can be obtained from $\mathcal{S}'$ by adding or removing at most a single event. Intuitively, differential privacy guarantees that an attacker with arbitrary background knowledge will not be able to (statistically) distinguish any computational result from $\mathcal{S}$ and $\mathcal{S}'$. This definition is formalized below.

Definition 1. (Event-Level Differential Privacy [9]) A randomized algorithm $\mathcal{K}$ gives event-level ϵ-differential privacy, if for any two neighboring streams $\mathcal{S}$ and $\mathcal{S}'$, and for any $O \subseteq Range(\mathcal{K})$,

$$P[\mathcal{K}(\mathcal{S}) \in O] \leq \exp(\epsilon) \cdot P[\mathcal{K}(\mathcal{S}') \in O],$$

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

Since two neighboring streams differ in a single event, it follows that an attacker is not able to distinguish the presence or absence of any event. Therefore, this privacy notion entitles *any* individual *plausible deniability* for *any* event in the published results. The privacy level is controlled by the parameter ϵ, also known as *privacy budget*. The smaller ϵ is, the more privacy we gain.

A standard technique to achieve differential privacy is the *Laplace mechanism* [8]. It injects properly calibrated Laplace noise into a function's output to mask the impact of any single tuple. The maximal impact of a tuple to the output of a function f is called its *sensitivity*.

Definition 2. (Sensitivity [8]) For a function $f : \mathcal{S} \to \mathbb{R}^d$, the sensitivity of f is $\Delta f = \max_{\mathcal{S},\mathcal{S}'} \|f(\mathcal{S}) - f(\mathcal{S}')\|_1$ for all neighboring streams $\mathcal{S}$ and $\mathcal{S}'$.

The Laplace noise needed to satisfy differential privacy is precisely calibrated by a function's sensitivity and the given privacy parameter, leading to the Laplace mechanism as follows.

THEOREM 1. [8] *For any function $f : \mathcal{S} \to \mathbb{R}^d$, the mechanism $\mathcal{K}$,*

$$\mathcal{K}(\mathcal{S}) = f(\mathcal{S}) + \left\langle \mathtt{Lap}_1(\frac{\Delta f}{\epsilon}), \ldots, \mathtt{Lap}_d(\frac{\Delta f}{\epsilon}) \right\rangle$$

gives ϵ-differential privacy, where $\mathtt{Lap}_i(\frac{\Delta f}{\epsilon})$ are i.i.d Laplace variables with scale parameter $\frac{\Delta f}{\epsilon}$.

For data streams, the graceful *composition properties* of differential privacy are particularly important for continuous privacy protection. The *sequential composition* property states that when combined differentially private subroutines still provide differential privacy guarantees.

THEOREM 2. [8] *Let each mechanism $\mathcal{K}_i$ provide ϵ_i-differential privacy. A sequence of $\mathcal{K}_i(D)$ over the database D provide $\sum \epsilon_i$-differential privacy.*

A special case is when the mechanisms are applied to *disjoint* databases, leading to the *parallel composition* property.

THEOREM 3. [8] *Let $\mathcal{K}_i$ provide ϵ_i-differential privacy. A sequence of $\mathcal{K}_i(D_i)$ over a set of disjoint databases D_i provides* $\max(\epsilon_i)$*-differential privacy.*

3.3 Problem Statement

In this paper, we consider the following problem: *Given an infinite data stream, at each discrete timestamp we would like to publish an accurate histogram that summarizes the event distribution in the current sliding window while satisfying event-level differential privacy.*

Following the convention in [18, 19], the accuracy (or utility) of a sanitized histogram is measured in terms of the expected *sum of squared error* (SSE). Given an original histogram $\mathbf{H}$ and its sanitized version $\widetilde{\mathbf{H}}$, the expected SSE is

$$\mathbb{E}(\mathtt{SSE}(\mathbf{H}, \widetilde{\mathbf{H}})) = \mathbb{E}\left(\sum_{i=1}^{|\mathbf{H}|}\left(H_i - \widetilde{H}_i\right)^2\right),$$

where H_i and $\widetilde{H}_i$ are the true and perturbed counts in $\mathbf{H}$ and $\widetilde{\mathbf{H}}$, respectively, and $|\mathbf{H}|$ is the number of bins. Accordingly, the error of a bin is measured by its expected squared error. In our problem, we publish a sanitized histogram at each timestamp summarizing the event distribution in the current sliding window, and therefore our utility goal is to minimize either the *average* or the *total* expected SSE of all published histograms.

4. TWO FIRST-CUT SOLUTIONS

In this section, we provide two first-cut solutions that help better understand the problem and motivate our solution.

4.1 Sliding Window Based Perturbation

To publish T histograms over a data stream[1], we can directly perturb the histograms of the sliding windows by the Laplace mechanism. The most straightforward approach is to make use of the sequential composition property to divide the total privacy budget ϵ into T portions, each being used for sanitizing a histogram. Since a single event can change exactly one bin's count by 1, the expected SSE of a histogram introduced by this approach is

$$\mathbb{E}(\mathtt{SSE}(\mathbf{H}, \widetilde{\mathbf{H}})) = \mathbb{E}\sum_{i=1}^{|\mathbf{H}|}\left(H_i - H_i - \mathtt{Lap}\left(\frac{T}{\epsilon}\right)\right)^2 = \frac{2|\mathbf{H}|T^2}{\epsilon^2}.$$

For an infinite data stream, we need to publish an infinite number of histograms (i.e., $T \to \infty$), and therefore the above method can barely provide useful results. However, if we observe that a single event can affect *at most* ω sliding windows that cover it, where ω is the sliding window size, we can do better. The sensitivity of all count queries needed to build all the histograms is ω because a single event can affect at most ω bin counts by 1. Therefore, the expected SSE of a histogram becomes $\frac{2|\mathbf{H}|\omega^2}{\epsilon^2}$, where $\omega \ll T$.

4.2 Time Unit Based Perturbation

Another alternative is to add Laplace noise to the counts of each time unit and generate the histogram of a sliding window by summing the counts of all time units it covers. Since a single event can affect *at most* one bin's count in *at most* one time unit by 1 (irrespective of the fact that the data stream is infinite), the sensitivity of all count queries is 1. Thus the expected SSE of the histogram corresponding to a time unit is $\frac{2|\mathbf{H}|}{\epsilon^2}$, leading to an expected SSE of $\frac{2|\mathbf{H}|\omega}{\epsilon^2}$ for a histogram of a sliding window with size ω. It can be seen that this method is strictly better than the sliding window based perturbation approach. Therefore, we consider the time unit based perturbation as the baseline approach (referred to as `Baseline` in the sequel) and propose novel techniques to further improve its utility.

5. OUR SOLUTION

In this section, we first provide an overview of our data stream sanitization framework and then elaborate its two key components.

5.1 Overview

Our retroactive-grouping-based framework (referred to as `RG`) consists of two novel modules: a sampling-based change monitoring module and a continuous histogram publication module. The former adaptively determines the best sampling rates to learn the underlying stream's evolution, based on which the latter takes proper publishing strategies to improve utility.

We first summarize the notations that are most frequently used in the sequel. We denote a time duration starting at timestamp t_i and ending at t_j by $[t_i, t_j]$, where $i < j$. The segment of the data stream $\mathcal{S}$ in $[t_i, t_j]$ is then denoted by $\mathcal{S}_{t_i}^{t_j}$. We denote the histogram corresponding to the time duration $[t_i, t_j]$ by $\mathbf{H}_{t_i,t_j}$ and the count of a bin B_l in $\mathbf{H}_{t_i,t_j}$ by H_{t_i,t_j}^l. The histogram published at time t_i for its sliding window's duration $[t_{i-\omega}, t_i]$ is written as $\mathbf{H}_{t_i}$ for short, and its count of a bin B_l is written as $H_{t_i}^l$. When the context is clear, we may omit some subscripts or superscripts. We use tilded versions to denote differentially private variables.

[1]Here we introduce the parameter T just for the theoretical analysis. Our solution supports infinite streams.

Algorithm 1 Private histogram publication at time t_i

```
Input: Data Stream $\mathcal{S}$
Input: Current timestamp $t_i$
Input: Privacy budgets $\epsilon_1$ and $\epsilon_2$ s.t. $\epsilon_1 + \epsilon_2 = \epsilon$
Input: Sliding window size $\omega$
Input: Histograms $\widetilde{\mathbf{H}}_{t_{k-1},t_k}$ with $i-\omega+1 \le k \le i-1$
Output: Sanitized histogram $\widetilde{\mathbf{H}}_{t_i}$
 1: Calculate the difference vector $\mathbf{d}$ between $[t_{i-1}, t_i]$ and
    $[t_{i-2}, t_{i-1}]$ using $\epsilon_1$;                    //see Algorithm 2
 2: for each bin $B_j$ do
 3:     if $\mathbf{d}[j] \ge \frac{2}{\epsilon_2^2}$ then      //see Section 5.3 for the use of $\frac{2}{\epsilon_2^2}$
 4:         $\widetilde{H}_{t_{i-1},t_i}^j = H_{t_{i-1},t_i}^j + \mathtt{Lap}\left(\frac{1}{\epsilon_2}\right)$;
 5:         $\mathcal{G}(B_j) = \varnothing$;
 6:     else
 7:         Add $H_{t_{i-1},t_i}^j$ to $\mathcal{G}(B_j)$;
 8:         if $|\mathcal{G}(B_j)| \ge m$ then            //see Section 5.3 for $m$
 9:             $\widetilde{H}_{t_{i-1},t_i}^j = \frac{\sum_{H_k \in \mathcal{G}(B_j)} H_k}{|\mathcal{G}(B_j)|} + \mathtt{Lap}\left(\frac{1}{|\mathcal{G}(B_j)|\epsilon_2}\right)$;
10:             $\mathcal{G}(B_j) = \varnothing$;
11:         else
12:             $\widetilde{H}_{t_{i-1},t_i}^j = \widetilde{H}_{t_{i-2},t_{i-1}}^j$;
13:     $\widetilde{H}_{t_i}^j = \sum_{k=i-\omega+1}^{i} \widetilde{H}_{t_{k-1},t_k}^j$;
14: return $\widetilde{\mathbf{H}}_{t_i}$;
```

At a timestamp t_i, Algorithm 1 presents how our framework releases a sanitized histogram for the current sliding window. Since we concern with event-level differential privacy, it is helpful to consider the events within different time units as *disjoint* databases. According to the parallel property, this allows us to use the full privacy budget on each time unit and offers us much flexibility in algorithm design. In Line 1, the monitoring module decides a proper sampling rate and calculates the difference vector $\mathbf{d}$ that measures the count differences of all bins between the current time unit and the previous time unit in terms of squared error. Since the monitoring module operates on the raw data, it needs to be guarded by differential privacy. For this purpose, the privacy budget ϵ_1 is assigned to it. Instead of comparing the SSE of the histograms in whole, we observe that the counts of different bins may change at different rates and therefore process each bin individually. Each element in $\mathbf{d}$ records the difference of the counts of a bin.

If the count difference of a bin is larger than the error introduced by the Laplace mechanism ($\frac{2}{\epsilon_2^2}$, see Section 5.3 for explanation), the best strategy is to employ the Laplace mechanism to generate the bin's noisy count (Line 4). Otherwise, there is room to improve. A natural idea is to approximate the current count with the previous (noisy) count, a method used in [13], which we call *passive publishing.* Unfortunately, in our setting passive publishing alone may not be able to improve accuracy. In [13], passive publishing benefits by saving on the privacy budget for future time units, while in our case each time unit can use the privacy budget in full. This implies that we can always use the Laplace mechanism to generate a noisy count with error no more than that of passive publishing, which is the sum of the Laplace error of the same magnitude and the approximation error (i.e., the actual difference between the current and the previous counts).

Here our key insight is that while not able to save on privacy budget, the time unit passively released (referred to as *passive time unit*) does help reduce sensitivity. To gain this benefit, we deliberately select some time units to passively publish and later group them with the time units with close counts, which we call *retroactive grouping*. At timestamp t_i, if we can find a retroactive group with a size $\geq m$ (a threshold determined by error analysis later) for a bin, we add *reduced* Laplace noise to the count (Line 9); otherwise we use passive publication (Line 12) to approximate the count. Finally, the noisy count of a bin in the current sliding window is the sum of the noisy counts of all time units in the window (Line 13).

5.2 Sampling-based Monitoring

Monitoring the evolution of a data stream is key to taking the proper publishing strategy in our framework. At each timestamp, we compute the differences between the counts of the current time unit and those of the previous time unit. *Sampling* is a common tool used on data streams mainly for the reason of efficiency. In the context of differential privacy, sampling also has an impact on the utility aspect. Previous research [15] has indicated the amplification effect of sampling on the privacy budget, which is formalized below.

THEOREM 4. [15] *Let $\mathcal{K}$ be an ϵ-differentially private algorithm. Let $\mathcal{K}_\gamma$ be another algorithm that first samples each tuple independently in its input dataset with probability γ and then applies $\mathcal{K}$ to the sampled dataset. $\mathcal{K}_\gamma$ satisfies $\ln(1+\gamma(e^\epsilon-1))$-differential privacy.*

Theorem 4 suggests that if the sample generated is *representative* (i.e., the statistics obtained over it are approximately the same as those of the original data), it is possible to obtain more accurate results over the sample by taking advantage of reduced Laplace noise. However, since sampling itself introduces a new source of error, *sampling error*, sampling may not always be helpful. To make a wise decision on whether or not to employ sampling (and the proper sampling rate), we need to carefully quantify the sampling error. While sampling has been used in previous research on differential privacy [12, 15], to our best knowledge, the sampling error was never quantified before. In this paper, we provide theoretical analysis on the sampling error.

Algorithm 2 presents the pseudocode of our sampling-based monitoring module. At each timestamp t_i, for each bin B_j, we estimate the difference between its counts in the current time unit $[t_{i-1}, t_i]$ and the previous time unit $[t_{i-2}, t_{i-1}]$. The first step is to determine the proper sampling rate γ_j for B_j (Line 2 of Algorithm 2). The sampling technique used in Theorem 4 is *Bernoulli sampling* [16]: For each coming event (of the population), perform a Bernoulli trial with the success probability $0 < \gamma_j \leq 1$, which is the sampling rate. If a success occurs, include the trial event as part of the sample; otherwise discard the event. Under Bernoulli sampling, given a sample size n_j, the population size N_j (i.e., B_j's true count in time unit $[t_{i-1}, t_i]$) can be estimated by $\hat{N}_j = \frac{n_j}{\gamma_j}$. $\hat{N}_j$ is an unbiased estimator of N_j. Since we use squared error to measure the utility of sanitized histograms, here we use mean squared error (MSE) to quantify the sampling error. Since the sample size n_j is a binomial variable, we have

Algorithm 2 Sampling-based monitoring at time t_i

```
Input: Data stream $\mathcal{S}$
Input: Current timestamp $t_i$
Input: Privacy parameter $\epsilon_1$
Input: Histogram $\widetilde{\mathbf{H}}_{t_{i-2},t_{i-1}}$
Output: Difference vector $\mathbf{d}$
1: for each bin $B_j$ do
2:     Compute the sampling rate $\gamma_j$;
3:     Generate $S_j$ by sampling events of $B_j$ in $\mathcal{S}_{t_{i-1}}^{t_i}$ with rate $\gamma_j$;
4:     $\widetilde{N}_j = \frac{|S_j| + \mathtt{Lap}\left(\frac{1}{\ln(e^{\epsilon_1}-1+\gamma_j)-\ln\gamma_j}\right)}{\gamma_j}$;
5:     $\mathbf{d}[j] = \left(\widetilde{H}^j_{t_{i-2},t_{i-1}} - \widetilde{N}_j\right)^2$;
6: return $\mathbf{d}$;
```

$$\mathbb{E}(\mathtt{MSE}(N_j, \hat{N}_j)) = \sum_{i=0}^{N_j} P(n_j = i)(\frac{n_j}{\gamma_j} - N_j)^2, \tag{1}$$

where $P(n_j = i) = \binom{N_j}{i}\gamma_j^i(1-\gamma_j)^{N_j-i}$ is the probability of having a sample size i. By solving Equation 1, we obtain $\mathbb{E}(\mathtt{MSE}(N_j, \hat{N}_j)) = \frac{N_j(1-\gamma_j)}{\gamma_j}$. This is the sampling error introduced in our monitoring module.

On the other hand, due to sampling, we enjoy reduced Laplace noise (see its construction in Line 4 of Algorithm 2), whose error can be quantified as per Theorem 4 as

$$\frac{2\gamma_j^2}{(\ln(e^{\epsilon_1}-1+\gamma_j)-\ln\gamma_j)^2}.$$

The total error is the sum of the sampling error and the Laplace error. Thus, we can express the total error as a function of γ_j and are interested in identifying the sampling rate γ_j that minimizes the total error, as formulated below:

$$\begin{aligned} \underset{\gamma_j}{\text{minimize}} \quad & \frac{N_j(1-\gamma_j)}{\gamma_j} + \frac{2\gamma_j^2}{(\ln(e^{\epsilon_1}-1+\gamma_j)-\ln\gamma_j)^2} \\ \text{subject to} \quad & 0 < \gamma_j \leq 1 \end{aligned}$$

Since γ_j is confined to the range $(0, 1]$, we can easily find the best rate by discretizing the range and applying the brute force method[2]. The above formulation is general to capture both the cases of sampling and *not* sampling. If $\gamma_j = 1$ is chosen, it means that no sampling should be applied. Therefore, our solution is general to make a decision on whether or not to apply sampling at all. Now the only unsolved problem is that in practice the population size N_j (and even its noisy version) is unknown before choosing γ_j. Here we estimate N_j by the noisy count in the previous time unit due to the fact that the statistics of a data stream may not change substantially in successive time units, a key assumption made in previous research [13]. Our experiments confirm that our foregoing error analysis precisely identifies

[2]In the experiments, we set the granularity to 1%.

the best sampling rate. In general, we observe that the benefit of sampling is more visible when the population size is relatively small or the privacy budget is small. This allows us to use a very small ϵ_1 (e.g., 0.1ϵ) to accurately monitor the evolution of a data stream.

For each bin B_j, after determining its sampling rate γ_j, we generate the sample S_j by including each event associated with B_j in the stream segment $\mathcal{S}_{t_{i-1}}^{t_i}$ with probability γ_j. We add reduced Laplace noise $\texttt{Lap}\left(\frac{1}{\ln(e^{\epsilon_1}-1+\gamma_j)-\ln\gamma_j}\right)$ to the true sample size $|S_j|$, estimate the count by the unbiased estimator introduced before, and record the squared error between this estimation and the noisy count of the previous time unit in the difference vector $\mathbf{d}$. Since we keep track of the changes of bins independently, the same procedure needs to be applied to all bins.

5.3 Continuous Histogram Publication

As analyzed in Section 4, the time unit based perturbation method achieves strictly better accuracy than the sliding window based perturbation method. Thus we design our continuous histogram publication module based on this conclusion. Depending on the output of the monitoring module, our framework takes three different strategies to calculate the bins' noisy counts in the current time unit $[t_{i-1}, t_i]$.

We first quantify the error of the Laplace mechanism and use it as the sentinel to indicate which strategy to take. Since any single event can affect only one bin's count by exactly 1 (i.e., the sensitivity of the corresponding count queries is 1) and ϵ_2 is reserved for publication, for any count H_j, the squared error of its noisy version generated by the Laplace mechanism is

$$\mathbb{E}\left(H_j + \texttt{Lap}\left(\frac{1}{\epsilon_2}\right) - H_j\right)^2 = \mathbb{E}\left(\texttt{Lap}\left(\frac{1}{\epsilon_2}\right)\right)^2 = \frac{2}{\epsilon_2^2}.$$

This explains the threshold used in Line 3 of Algorithm 1. If the difference of a bin B_j's counts between the current time unit and the previous unit is $\geq \frac{2}{\epsilon_2^2}$, it is clear that using the standard Laplace mechanism to perturb the current count H_{t_{i-1},t_i}^j is the best strategy. Recall that we can use ϵ_2 to perturb the count and that the sensitivity is 1, and thus the Laplace noise is generated with scale parameter $\frac{1}{\epsilon_2}$. We will formally analyze the privacy guarantee of our solution in Section 6.2.

If the difference is $< \frac{2}{\epsilon_2^2}$, we examine whether it is possible to benefit from retroactive grouping and take the corresponding strategies. In the context of data streams, *the grouping must be done in a way that does not increase the sensitivity*; otherwise its benefit will be canceled out due to the increased sensitivity. It follows that the count of a bin in a time unit that has been previously released by the Laplace mechanism (using ϵ_2[3]) should never be asked again. For this reason, we deliberately choose some time units for passive publishing (and thus their counts have not been answered by the Laplace mechanism using ϵ_2) and later group a bin's current time unit with its previously passively published time units for reduced error.

We denote the retroactive group of a bin B_j by $\mathcal{G}(B_j)$ and its size by $|\mathcal{G}(B_j)|$. It contains the counts of the preceding passive time units up to (but not including) the most recent non-passive time unit. After each non-passive release, we empty $\mathcal{G}(B_j)$ (Lines 5 and 10 of Algorithm 1). It can be seen that such a construction does not increase the sensitivity. As a result, the counts of the first $|\mathcal{G}(B_j)| - 1$ members in $\mathcal{G}(B_j)$ are approximated by the noisy count of the most recent non-passive time unit, and the last member, the count of the current time unit H_{t_{i-1},t_i}^j, will enjoy reduced Laplace noise. More specifically, H_{t_{i-1},t_i}^j is estimated by adding $\texttt{Lap}\left(\frac{1}{|\mathcal{G}(B_j)|\epsilon_2}\right)$ to the count average of the retroactive group, that is

$$\widetilde{H}_{t_{i-1},t_i}^j = \frac{\sum_{H_k\in\mathcal{G}(B_j)} H_k}{|\mathcal{G}(B_j)|} + \texttt{Lap}\left(\frac{1}{|\mathcal{G}(B_j)|\epsilon_2}\right).$$

We give the sensitivity of the group average here and prove its privacy guarantee later.

THEOREM 5. *The sensitivity of* $\frac{\sum_{H_k\in\mathcal{G}(B_j)} H_k}{|\mathcal{G}(B_j)|}$ *is* $\frac{1}{|\mathcal{G}(B_j)|}$.

PROOF. Since a single event can change a bin's count in at most one time unit by 1, it can change $\sum_{H_k\in\mathcal{G}(B_j)} H_k$ by at most 1, and therefore can change the average by at most $\frac{1}{|\mathcal{G}(B_j)|}$. This establishes the proof. □

For static histograms, we always prefer a larger group provided that the approximation error is low (i.e., all counts in the group are close) because the Laplace noise reduction of *every* member in the group is proportional to the group size. Unfortunately, this is not the case for a data stream. We need more sophisticated design. For a data stream, it is mandatory to release a histogram at each timestamp and the histogram is consumed in real time. The previously published counts (i.e., the first $|\mathcal{G}(B_j)| - 1$ members in $\mathcal{G}(B_j)$) cannot benefit from the reduced Laplace noise due to retroactive grouping. This fact requires to carefully select the retroactive group size, the parameter m in Line 8 of Algorithm 1.

As per the problem definition, for a bin B_j, our goal is to minimize the average expected squared error of its retroactive group. To this end, we would like to express the average error as a function of m in order to identify the best m value. Here we need to first understand the squared error of a passive publication. Let H_i be a count from which H_j is approximated by a passive publication. Note that H_i and H_j do not have to belong to two consecutive time units because if the data stream evolves very slowly, H_j could be well approximated by the count of a time unit long ago. In a passive publication, we approximate H_j with the noisy version $\widetilde{H}_i$ of H_i and thus the expected error is

$$\begin{aligned}\mathbb{E}\left((H_j - \widetilde{H}_i)^2\right) &= \mathbb{E}\left((H_j - H_i - \texttt{Lap}(\lambda))^2\right)\\ &= \underbrace{(H_j - H_i)^2}_{\text{Approximation error}} + \underbrace{2\lambda^2}_{\text{Laplace error}}\end{aligned}$$

where λ is the scale parameter of the Laplace noise added to $\widetilde{H}_i$. λ may not always be $\frac{1}{\epsilon_2}$ as $\widetilde{H}_i$ may be generated with reduced Laplace noise by retroactive grouping. The first item $(H_j - H_i)^2$ represents the *approximation error*, while the second item $2\lambda^2$ represents the *Laplace error*. The trade-off between these two types of errors lies at the core of our approach: *we would like to trade a small approximation error for a large reduction of Laplace error.* If the standard

[3]There is no need to worry about the queries issued using ϵ_1. This point will be made clear in Section 6.2.

Laplace mechanism is applied to H_i (in this case $\lambda = \frac{1}{\epsilon_2}$), passive publishing cannot perform better than directly applying the Laplace mechanism to H_j because $(H_j - H_i)^2 \geq 0$. Now we are ready to present the average expected squared error of noisy counts in a retroactive group.

THEOREM 6. *For a retroactive group $\mathcal{G}(B_j)$ of size m, its average expected squared error is*

$$\frac{\sum_{i=1}^{m-1}(H_i - H_0)^2 + (H_m - \bar{H})^2}{m} + \frac{2(m-1)\lambda^2}{m} + \frac{2}{(m\epsilon_2)^2},$$

where H_i with $1 \leq i \leq m$ is the ith element in $\mathcal{G}(B_j)$, $\bar{H}$ is the average of H_i's , H_0 is the true count from which the first $m-1$ members in $\mathcal{G}(B_j)$ are approximated, and λ is the scale of the Laplace noise added to H_0.

PROOF. Since the first $m-1$ members are passively released, all their noisy counts are approximated by $H_0 + \texttt{Lap}(\lambda)$. Then, by definition, the average expected squared error of a retroactive group is

$$\mathbb{E}\left(\frac{\sum_{i=1}^{m}(\widetilde{H}_i - H_i)^2}{m}\right)$$

$$= \frac{1}{m}\mathbb{E}\left(\sum_{i=1}^{m-1}(\widetilde{H}_i - H_i)^2 + \left(H_m - \frac{\sum_{i=1}^{m} H_i}{m} - \texttt{Lap}\left(\frac{1}{m\epsilon_2}\right)\right)^2\right)$$

$$= \frac{1}{m}\left(\sum_{j=1}^{m-1}(H_j - H_0)^2 + 2(m-1)\lambda^2 + (H_m - \bar{H})^2 + \frac{2}{m\epsilon_2^2}\right)$$

$$= \frac{\sum_{j=1}^{m-1}(H_j - H_0)^2 + (H_m - \bar{H})^2}{m} + \frac{2(m-1)\lambda^2 + \frac{2}{m\epsilon_2^2}}{m},$$

where the first item represents the approximation error and the second item represents the Laplace error. □

If we only consider the Laplace error, it is easy to derive the best value of m:

$$m = \underset{k \in \mathbb{N}^+ \setminus \{1\}}{\operatorname{argmin}} \left(\frac{2(k-1)\lambda^2}{k} + \frac{2}{(k\epsilon_2)^2}\right),$$

which gives $m = \frac{2}{(\lambda\epsilon_2)^2}$. However, with the increase of m, the approximation error often also increases, which requires to select a smaller m value. Since the approximation error is data-dependent and difficult to quantify, we empirically choose $m = \frac{2}{\lambda\epsilon_2}$ to account for the approximation error, which works well for all experimental datasets. Since λ is a variable that changes with the data stream, the value of m also changes.

We illustrate how to apply different publishing strategies in the following example.

EXAMPLE 2. *Consider the data stream in Figure 2. We focus on a single bin with its counts given on top of the time units. At timestamp t_2, since it is the first time unit, its true count 5 is perturbed by the Laplace mechanism. At t_3, let the difference between 75 and 5 detected by the monitoring module is $\geq \frac{2}{\epsilon_2^2}$, the true count 75 is released by the Laplace mechanism. Let the differences calculated from t_4 to t_9 be all $< \frac{2}{\epsilon_2^2}$. At t_4, the true count 74 in $[t_3, t_4]$ is passively released because the current retroactive group size is $< m = 2$. At t_5, the true count 73 is published using retroactive grouping and the group is then emptied. After that, the counts at t_6, t_7 and t_8 are passively released because now $m = 4$. Finally the true count 75 in $[t_8, t_9]$ is released using retroactive grouping.*

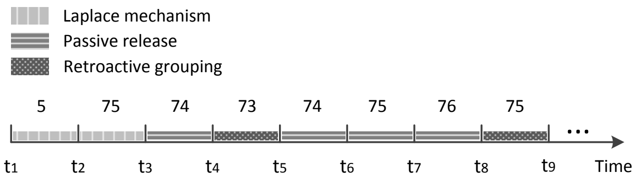


Figure 2: An illustration of continuous histogram publication

After obtaining a bin's noisy count in the current time unit using one of the publishing strategies, its noisy count in the current sliding window is the sum of the noisy counts of the time units covered by the sliding window (Line 13 of Algorithm 1). We repeat this procedure for all bins to produce the histogram to publish at t_i.

6. THEORETICAL ANALYSIS

In this section, we provide the theoretical analysis of RG.

6.1 Utility Analysis

We give the major utility result in the following theorem.

THEOREM 7. *For (a segment of) a data stream with T time units, if $T = 2^{l+1} - 1$ for some $l \in \mathbb{N}^+$, the minimum average expected SSE of the histograms published by RG is*

$$\frac{\omega|\mathbf{H}|}{3\epsilon_2^2} \cdot \frac{28(T+1)^2 - 60T - 28}{(T+1)^3 - (T+1)^2};$$

otherwise its minimum average expected SSE is

$$\frac{2\omega|\mathbf{H}|}{3T\epsilon_2^2} \cdot \left(14 - \frac{42}{2^{k+1}} + \frac{16 + 12T}{2^{2k+2}} + \frac{3}{T - 2^{k+1} + 1}\right),$$

where $k = \lfloor \log_2(T+1) - 1 \rfloor$.

PROOF. It is easy to see that the minimum average expected SSE is achieved when the events in each bin arrive at a fixed speed. In this case, the approximation error is 0 and the SSE solely comes from the Laplace error. By design, for any bin, its first time unit's count will be published by the Laplace mechanism with the Laplace error $\frac{2}{\epsilon_2^2}$. After that, if $T = 2^{l+1} - 1$ for some $l \in \mathbb{N}^+$, the retroactive group sizes are $2^1, 2^2, \cdots, 2^l$; otherwise, the retroactive group sizes are $2^1, 2^2, \cdots, 2^k, T - \sum_{i=0}^{k} 2^i$, where $k = \lfloor \log_2(T+1) - 1 \rfloor$. For both cases, the λ value of the ith group is $\frac{2}{2^i\epsilon_2}$. For space reasons, we only prove the latter case. The former is a simplified case of the latter and can be similarly proved. By Theorem 6, the Laplace error of the ith retroactive group is $\frac{2 \cdot 2^2(2^i - 1)}{(2^i\epsilon_2)^2} + \frac{2}{2^i\epsilon_2^2}$, where $1 \leq i \leq k$, and the Laplace error of the last group is $\frac{2 \cdot 2^2(T - \sum_{i=0}^{k} 2^i - 1)}{(2^{k+1}\epsilon_2)^2} + \frac{2}{(T - \sum_{i=0}^{k} 2^i)\epsilon_2^2}$. Then we can express the minimum average expected squared error of a bin in a time unit as

$$\frac{\sum_{i=0}^{k}\left(\frac{2 \cdot 2^2(2^i-1)}{(2^i\epsilon_2)^2} + \frac{2}{2^i\epsilon_2^2}\right) + \frac{2 \cdot 2^2(T - \sum_{i=0}^{k} 2^i - 1)}{(2^{k+1}\epsilon_2)^2} + \frac{2}{(T - \sum_{i=0}^{k} 2^i)\epsilon_2^2}}{T}$$

$$= \frac{2}{T\epsilon_2^2} \cdot \left(5\sum_{i=0}^{k}\frac{1}{2^i} - 4\sum_{i=0}^{k}\frac{1}{2^{2i}} + \frac{4(T - 2^{k+1})}{2^{2k+2}} + \frac{1}{T - 2^{k+1} + 1}\right)$$

$$= \frac{2}{3T\epsilon_2^2} \cdot \left(14 - \frac{42}{2^{k+1}} + \frac{16 + 12T}{2^{2k+2}} + \frac{3}{T - 2^{k+1} + 1}\right)$$

Since each histogram covers ω time units and has $|\mathbf{H}|$ bins, the minimum average expected SSE becomes

$$\frac{2\omega|\mathbf{H}|}{3T\epsilon_2^2}\cdot\left(14-\frac{42}{2^{k+1}}+\frac{16+12T}{2^{2k+2}}+\frac{3}{T-2^{k+1}+1}\right).$$

This completes the proof. □

Theorem 7 can be used to analyze the utility of not only an entire data stream (even an infinite stream by setting $T\to\infty$), but also a segment of a data stream. Since the minimum SSE is achieved when the approximation error is 0, Theorem 7 helps to identify the maximum Laplace error reduction that can be achieved by RG. For example, when $T=63$, the minimum Laplace error is $0.14\cdot\frac{\omega|\mathbf{H}|}{\epsilon_2^2}$. When $\epsilon_2=0.9\epsilon$, which is our experimental setting, the minimum Laplace error is just 8.6% of the Laplace error introduced by `Baseline` (which is $\frac{2\omega|\mathbf{H}|}{\epsilon^2}$).

When analyzing the utility of RG, we cannot ignore the approximation error, which is another source of error in RG. Due to the data-dependent nature of RG, it is very difficult to precisely quantify the approximation error. But it is helpful to consider the Laplace error reduction as the tolerance of RG for the approximation error. As long as the approximation error is less than the reduction, it is beneficial to use RG. In Section 7, we experimentally show that this is often the case on real-life data.

6.2 Privacy Analysis

To establish the privacy guarantee of RG, we mainly rely on the two composition properties of differential privacy. Now we give the formal privacy guarantee of RG.

THEOREM 8. RG *is ϵ-differentially private.*

PROOF. By sequential composition property, to prove that RG is ϵ-differentially private, it suffices to independently prove that the sampling-based change monitoring module satisfies ϵ_1-differential privacy and that the continuous histogram publication module satisfies ϵ_2-differential privacy, where $\epsilon=\epsilon_1+\epsilon_2$.

For the sampling-based change monitoring module, since any single event can appear in exactly one time unit, by definition of event-level differential privacy, we can consider the events within different time units as *disjoint* databases no matter how many time units there are in the data stream. By the parallel composition property, if the procedure for processing each time unit is ϵ_1-differentially private, the monitoring module for the entire data stream is still ϵ_1-differentially private. For any time unit, since an event is associated with exactly one bin, we can again apply the parallel composition property, which requires to prove that the noisy count of each bin in the time unit is ϵ_1-differentially private. Note that since the output of the monitoring module is based on raw data, it needs to be properly sanitized even though it does not directly publish its output.

For any bin B_j, we first prove that without sampling adding $\mathtt{Lap}\left(\frac{1}{\ln(e^{\epsilon_1}-1+\gamma_j)-\ln\gamma_j}\right)$ satisfies $(\ln(e^{\epsilon_1}-1+\gamma_j)-\ln\gamma_j)$-differential privacy. This directly follows the fact that the sensitivity of the count query is 1. Note that our method of learning γ_j does not consume any privacy budget as it is based on differentially private results. Now by Theorem 4, we learn that the sampling-based mechanism satisfies ϵ_1-differential privacy because

$$\ln(1+\gamma_j(e^{\ln(e^{\epsilon_1}-1+\gamma_j)-\ln\gamma_j}-1))=\epsilon_1.$$

Table 1: Dataset statistics

Dataset	Event no.	$\|\mathbf{H}\|$	Timestamp no.
World Cup	1,352,804,107	88,762	[176, 2112]
Traffic	8,107,643	342	[168, 1008]

Next we prove that the continuous histogram publication module is ϵ_2-differentially private. Establishing the privacy guarantee of the publication module is more tricky as we issue queries that span over several time units, and therefore we cannot treat the events in different time units as disjoint databases. But the fact that an event can fall into exactly one bin still holds. Therefore, by parallel composition, it suffices to prove that all noisy answers to all queries over any bin gives ϵ_2-differential privacy. We establish the proof by transforming all these queries into a standard Laplace mechanism. For each bin, we ask two types of queries: in Line 4 of Algorithm 1, we issue a count query to obtain the noisy count in the current time unit; in Line 9, we issue a query to obtain the noisy average of a retroactive group. By Theorem 5, the latter is equivalent to a sum query: what is the sum of the counts in a retroactive group, as the group size can be learned from the differentially private results. By design, no event can be involved in more than one of these queries because no time unit is involved in more than one query. It can be verified that for all these count and sum queries, a single event can change only one of the answers by exactly 1. Therefore, due to Theorem 1, the publication module is ϵ_2-differentially private. □

6.3 Space and Time Complexity

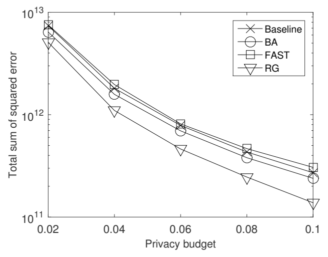
For real-time stream processing, the space complexity and time complexity of an algorithm are very important factors to consider. For the space complexity, RG never requires to store the entire data stream. At any timestamp, it only needs to store the events in the current time unit, the noisy counts of the past $\omega-1$ time units and the retroactive groups for all bins. Let n be the number of events in the current time unit and m_{max} be the maximum group size over the data stream. It is easy to verify that the space complexity is $O(n+(m_{max}+\omega)|\mathbf{H}|)$. In practice, m_{max} and ω are normally small numbers (compared to n and $|\mathbf{H}|$).

The time complexity of RG for each publication can be derived from Algorithm 1. In the monitoring module, for each bin, we calculate its sampling rate, where the runtime cost is $O(1)$. For each coming event we perform a Bernoulli trial to decide whether to include it in the sample. Therefore, the time complexity of the monitoring module is $O(n+|\mathbf{H}|)$, where n is the number of events arriving in the current time unit and $|\mathbf{H}|$ is the number of bins. In the publication module, in the worst case, for each bin, we need to sum all elements in its retroactive group and sum the noisy counts of the ω time units in the sliding window. The time complexity of processing all bins is $O\left((m_{max}+\omega)|\mathbf{H}|\right)$. The total time complexity is then $O\left(n+(m_{max}+\omega+1)|\mathbf{H}|\right)$.

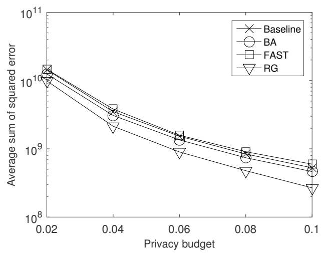
As such, our sanitization framework introduces only a small space and time overhead and is suitable for processing large-scale real-time data streams.

7. EXPERIMENTAL EVALUATION

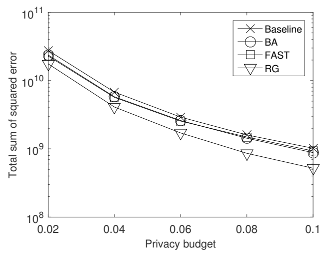
In this section, we empirically study the utility and scalability of our solution RG. For comparison reasons, we adapt the state-of-the-art techniques to our problem setting, including FAST [10] and BA [13]. We also implement `Baseline`



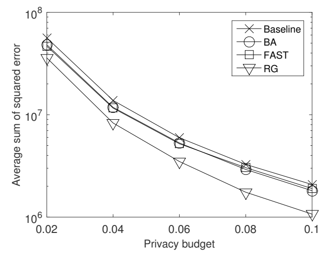
(a) Total SSE on World Cup



(b) Avg SSE on World Cup

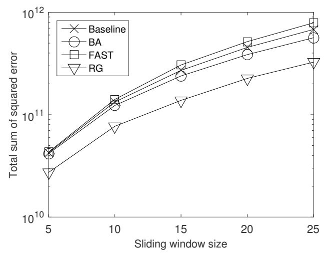


(c) Total SSE on Traffic

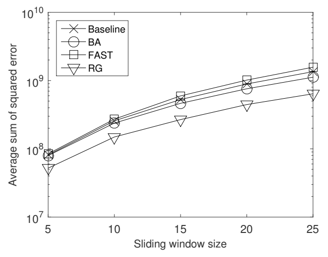


(d) Avg SSE on Traffic

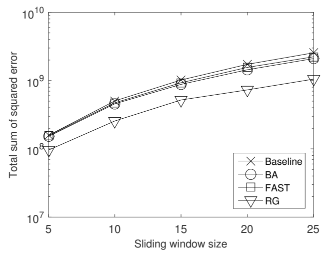
Figure 3: Error vs. privacy budget



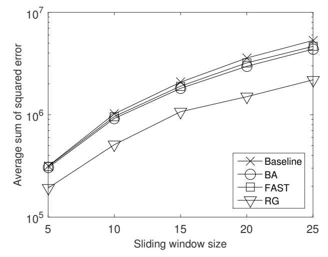
(a) Total SSE on World Cup



(b) Avg SSE on World Cup



(c) Total SSE on Traffic



(d) Avg SSE on Traffic

Figure 4: Error vs. sliding window size

introduced in Section 4.2. All methods are implemented in Java and tested on a machine with an Intel i7 2.40GHz CPU and 8GB RAM. In each experiment, every algorithm is executed 10 times, and the average is reported.

We use two real datasets in our experiments. The World Cup dataset[4] contains 1,352,804,107 requests made to the 1998 World Cup Web site in 88 consecutive days. After removing some irregular URLs, it contains 88,762 unique URLs, each being considered a bin. The Traffic dataset [7] contains passengers' 8,107,643 visits to 342 stations in the Montreal transportation system within one week. Similar to [13], we consider the number of timestamps as a variable parameter for investigation. The statistics of the datasets are summarized in Table 1. Unless explicitly specified, the default values of the parameters are set as follows: $\epsilon = 0.1$, $\epsilon_1 = 0.1\epsilon$, $\epsilon_2 = 0.9\epsilon$, $w = 15$, and the numbers of timestamps are 528 for World Cup and 504 for Traffic, respectively.

7.1 Utility Results

In our first set of experiments, we study the effect of privacy budget on all approaches. The total and average SSEs under different privacy budgets are reported in Figure 3. Note that the y-axis has a logarithmic scale. In general, the error decreases with the increase of privacy budget. As can be observed, RG substantially outperforms all competitors in all cases. RG can improve the accuracy of the second best approach by up to 43%.

Figure 4 presents the errors under different sliding window sizes. With the increase of the sliding window size, the error increases. Since all these approaches are essentially time unit based perturbation, including more time units in a sliding window results in a larger error. Also for this reason, the improvement of RG becomes more obvious with the increase of the sliding window size.

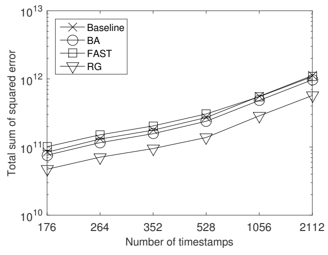
In the next set of experiments, we study the effect of the number of timestamps in Figure 5. This is equivalent to examining the effect of the magnitude of changes over time.

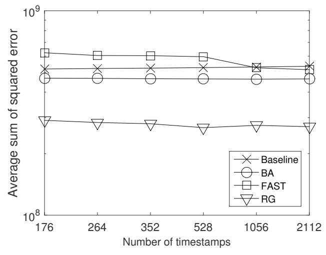
[4]Publicly available at: `http://ita.ee.lbl.gov/html/contrib/WorldCup.html`

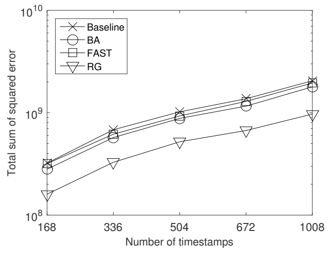
Since more timestamps means more publications, the total SSE increases accordingly. More interesting observations are from the average SSE. The average SSE of `Baseline` is quite stable under different numbers of timestamps. This is because the Laplace noise injected by `Baseline` is independent of the underlying data (i.e., it is calibrated by the sensitivity and the privacy budget, not the underlying data). In contrast, the other approaches all benefit from the increase of the number of timestamps or equivalently the reduced changes between consecutive time units. In particular, for RG, having more timestamps implies a slower evolution, which in turn entitles more chances of applying retroactive grouping to reduce noise. Again, in all cases, RG performs significantly better than the other approaches.

7.2 Scalability Results

According to our theoretical analysis, the space complexity of RG is very small. Due to the space limit, we only report the results on the time complexity. We study the runtime of RG w.r.t the two dominating factors, namely the number of events and the number of bins. Figure 6 shows the runtimes of different approaches under different numbers of events. To form the test datasets, we uniformly select a percentage of total events at random. The runtime of `Fast` is too large to fit into the figures and therefore is omitted. The units of the y-axis are *millisecond* and *microseconds* in Figure 6(a) and (b), respectively. While larger than `Baseline` and `BA`, the runtime of RG is in general negligible: in the worst case the runtime per publication is less than 700 milliseconds on World Cup and less than 1 millisecond on Traffic. Also, the runtime of RG grows linearly with the number of events, conforming to our theoretical analysis. The runtimes for different bin numbers are shown in Figure 7. Again the results of `FAST` are omitted for the same reason. Similarly, we generate the test datasets by randomly extracting a percentage of bins. We can observe the expected trends on both World Cup and Traffic: the runtime is very small and increases linearly with the number of bins.


(a) Total SSE on World Cup


(b) Avg SSE on World Cup


(c) Total SSE on Traffic

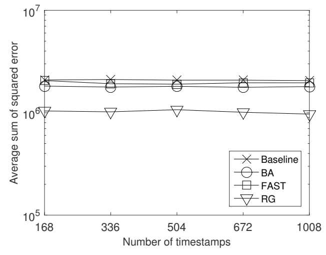
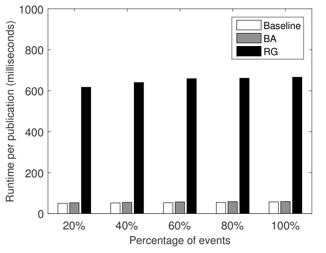

(d) Avg SSE on Traffic

Figure 5: Error vs. number of timestamps


(a) Runtime on World Cup

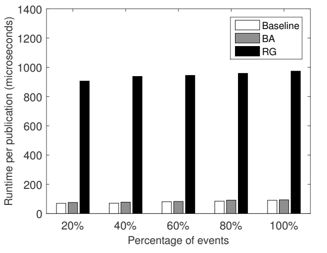
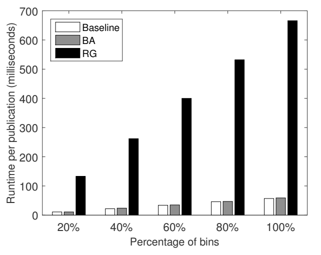

(b) Runtime on Traffic

Figure 6: Runtime vs. number of events

Finally, we would also like to note that since RG processes each bin independently, we can easily and substantially improve its runtime by parallel computing techniques.

8. CONCLUSION

In this paper, we consider the problem of continuously publishing histograms over an infinite data stream while satisfying differential privacy. We propose a novel sanitization framework consisting of two modules. The sampling-based monitoring module performs Bernoulli sampling by carefully quantifying the sampling error; the continuous publication module is equipped with a novel retroactive grouping technique to effectively reduce error. Our solution incurs a very small space and time overhead. Experiments over real datasets show that our framework outperforms the state-of-the-art competitors.


(a) Runtime on World Cup

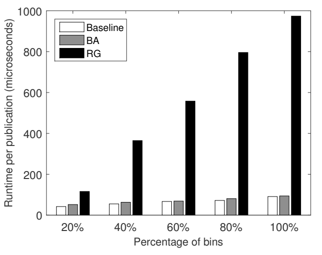

(b) Runtime on Traffic

Figure 7: Runtime vs. number of bins

9. REFERENCES

[1] G. Acs, C. Castelluccia, and R. Chen. Differentially private histogram publishing through lossy compression. In *ICDM*, 2012.

[2] C. C. Aggarwal. *Data Streams: Models and Algorithms*. Springer, 2007.

[3] J. Bolot, N. Fawaz, S. Muthukrishnan, A. Nikolov, and N. Taft. Private decayed predicate sums on streams. In *ICDT*, 2013.

[4] J. Cao, X. Qian, G. Ghinita, N. Li, E. Bertino, and K.-L. Tan. Efficient and accurate strategies for differentially-private sliding window queries. In *EDBT*, 2013.

[5] T.-H. H. Chan, M. Li, E. Shi, and W. Xu. Differentially private continual monitoring of heavy hitters from distributed streams. In *PETS*, 2012.

[6] T.-H. H. Chan, E. Shi, and D. Song. Private and continual release of statistics. *TISSEC*, 14(3):6, 2011.

[7] R. Chen, G. Acs, and C. Castelluccia. Differentially private sequential data publication via variable-length n-grams. In *CCS*, 2012.

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

[9] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *STOC*, 2010.

[10] L. Fan and L. Xiong. An adaptive approach to real-time aggregate monitoring with differential privacy. *TKDE*, 26(9):2094–2106, 2014.

[11] A. Friedman, I. Sharfman, D. Keren, and A. Schuster. Privacy-preserving distributed stream monitoring. In *NDSS*, 2014.

[12] G. Kellaris and S. Papadopoulos. Practical differential privacy via grouping and smoothing. *PVLDB*, 6(5):301–312, 2013.

[13] G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias. Differentially private event sequences over infinite streams. *PVLDB*, 7(12):1155–1166, 2014.

[14] C. Li, M. Hay, G. Miklau, and Y. Wang. A data- and workload-aware query answering algorithm for range queries under differential privacy. *PVLDB*, 7(5):341–352, 2014.

[15] N. Li, W. H. Qardaji, and D. Su. On sampling, anonymization, and differential privacy or, k-anonymization meets differentail privacy. In *ASIACCS*, 2012.

[16] C.-E. Sarndal, B. Swensson, and J. Vretman. *Model Assisted Survey Sampling*. Springer, 2003.

[17] N. Wang, J. Grossklags, and H. Xu. An online experiment of privacy authorization dialogues for social applications. In *CSCW*, 2013.

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

[19] X. Zhang, R. Chen, J. Xu, X. Meng, and Y. Xie. Towards accurate histogram publication under differential privacy. In *SDM*, 2014.