SPAS: Continuous Release of Data Streams under w-Event Differential Privacy

Abstract

Continuous release of data streams is frequently used in numerous applications. However, when data is sensitive, this poses privacy risks. To mitigate this risk, efforts have been devoted to devising techniques that satisfy a formal privacy notion called w-event differential privacy. Nevertheless, a recent benchmark reveals that none of the existing works offer a universally effective solution across all types of data streams, making it challenging to select an appropriate scheme for unknown data streams in practical scenarios.

We identify that all existing methods are *heuristic-based* and make *data-independent* decisions. In this paper, we change this landscape by introducing SPAS that is built on *data-dependent strategies*. Specifically, SPAS continuously predicts an optimal publishing strategy that minimizes the error of the released results based on the characteristics of the data stream. Additionally, we develop a weighted sparse vector technique to control data sampling and manage privacy budget consumption following an optimal publishing strategy. Comprehensive experimental evaluations demonstrate the efficacy of SPAS in adapting to diverse one-dimensional and multi-dimensional data streams for both data release and range query tasks. Our code is anonymously open-sourced via the link[1].

ACM Reference Format:

. 2018. SPAS: Continuous Release of Data Streams under w-Event Differential Privacy. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 14 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Continually observing and releasing valuable information from infinite streaming data are prevalent. For instance, the Water and Environmental Research Systems Network requires continuous observation of streaming sensor data to carry out anomaly detection in the environment [30]; federal and state transportation agencies observe streaming traffic data to alert drivers about congestion and accidents and plan new road pavements to accommodate predicted traffic loads [53]; banking institutions monitor consumers' financial transactions to detect unauthorized activities [22]. Since the streaming data may contain sensitive information of individuals, directly processing and releasing the statistical results over the data streams may leak personal privacy. For example, smart meters allow a malicious party to glean detailed household activity information from real-time fine-grained usage measurements [40].

Differential privacy (DP) [17] is a standard privacy definition for statistical analysis on databases containing sensitive user information, and is widely accepted by both academia and industry [14, 21, 23]. The impact of any single data in the database on the final statistical results can be limited by e^{ϵ}, where ϵ is the privacy budget and the smaller ϵ means stronger privacy protection.

When addressing the privacy issue of continuous data release, existing works [18, 42, 51] often prioritize event-level DP, which protects privacy of a single event (corresponds to a data value in the context of streaming data release). A stronger notion is user-level DP, which protects an individual's data throughout the stream, but this contradicts the goal of continuously publishing data and getting useful results in an infinite stream. Between event-level and user-level DP, in this paper, we investigate an under-investigated notion called w-event DP [31], which provides an ϵ-DP guarantee for all events within a *sliding window* of length w, which is reasonable in many real-world scenarios. That is, if a sensitive behavior is associated with a series of events, protecting a sequence is semantically more reasonable than protecting a single event.

Several existing works have been proposed [11, 24, 31, 34, 41, 50, 52] to publish infinite streams in w-event DP. However, a recent benchmark [46] of all these existing works reveals that no scheme can perform consistently well across all different types of real-world data streams used for evaluation. The study primarily attributes the reason for this performance inconsistency to their sampling processes (e.g., uniformly sample timestamps within a length-w window for publishing), which are either not data-dependent or too sensitive to data changes [46]. We would like to emphasize that another important factor is existing methods only prioritize/optimize for current timestamps and thus are short-sighted. Consequently, existing solutions either struggle to identify truly valuable information for release, or when they do identify valuable information, the most privacy budget may already be consumed.

Our Contributions. The above analysis delineated that the key factors hindering the existing data stream releasing schemes from dynamically adapting well to all data streams are data-independent sampling and data-independent privacy budget allocation Furthermore, addressing these issues cannot rely solely on the value of releasing current data, it also necessitates planning for the release of future data within the sliding window that contain the current timestamp. Drawing from this insight, we present a novel scheme SPAS (Stream Publishing based on Adaptive SVT), tailored for releasing infinite data streams under w-event DP. We summarize our contributions as follows.

Firstly, we propose a method for predicting a data-releasing strategy to guide data sampling and privacy budget allocation. We predict an optimal sampling count within the entire window starting from the current timestamp based on the analysis of historical publications. Specifically, we formulate an error function for the

[1] https://anonymous.4open.science/r/Sliding-window-data-release-revision-2BB7/

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

expected error of the releasing results, and then obtain the optimal sampling count by solving an optimization problem. By controlling the number of sampled data for release to approach the optimal sampling count, we can achieve the results' accuracy to the theoretically optimal accuracy. Meanwhile, knowing the approximate number of sampled data can help plan a reasonable privacy budget allocation to avoid insufficient utilization or premature depletion of the privacy budget.

Secondly, we design a weighted sparse vector technique (SVT) as the controller for adaptive data sampling. In this paper, the data sampling controller needs to sample the valuable data for release while ensuring that the consumption of the privacy budget within the current window does not exceed the total privacy budget. SVT [37] is a commonly used DP technique for controlling data sampling based on certain conditions and total privacy budget. However, existing SVT cannot handle the data with different weights, i.e., releasing each data consumes a different amount of privacy budget. Since the optimal sampling count varies with changes in the data stream, the allocated privacy budget for sampled data may also change. Therefore, we design a weighted SVT and ensure that it still provides the same privacy guarantee as the original SVT.

Thirdly, we provide parameter details of SPAS for deployment in practical scenarios, along with further optimization for privacy budget allocation. Specifically, we give the method to determine the initial optimal sampling count within the beginning window of the data stream where no historical releasing results are available. Then, we explain how to update the key parameter for calculating the optimal sampling count based on historical releasing results during the practical releasing process. Additionally, we further maximize the utilization of the privacy budget by avoiding unnecessary privacy budget consumption during the sampling process and ensuring the actual number of sampled data within each window reaches the optimal sampling count.

Finally, we conduct a comprehensive experimental evaluation. We conduct experiments on six real-world datasets and three synthetic datasets, including one-dimensional and multi-dimensional data streams, demonstrating the effectiveness of SPAS. When compared to seven well-established schemes, SPAS consistently ranks within the top three in terms of accuracy across all data streams for both releasing tasks and range query tasks. In contrast, other schemes occasionally perform poorly on certain streams due to their lack of adaptability. The evaluation results reveal the powerful generalization of SPAS, making it suitable for addressing the stream-releasing challenges in real-world applications.

2 Preliminary

2.1 Problem Statement

Assume that there is a dynamic dataset and that we need to release a snapshot of this dataset at each timestamp while preserving data privacy. Each snapshot is a histogram computed from the dynamic dataset, which can be denoted as H_i at the i^{th} timestamp. Each bin of H_i is a count value for an attribute of the dynamic dataset and we assume that the attributes of the dynamic dataset remain unchanged during the whole release process. For example, if the dynamic dataset continuously records daily transactions of a store, each bin of H_i on the i^{th} day can be the sales quantity of each item.

More specifically, there is a sensitive data stream $\mathcal{H} = \{H_1, \cdots H_i\}$. We need to release a data stream $\hat{\mathcal{H}} = \{\hat{H}_1, \cdots, \hat{H}_i\}$, where $\hat{H}_i$ is the noisy version of H_i at the i^{th} timestamp. The goal is to make $\hat{\mathcal{H}}$ close to $\mathcal{H}$ while satisfying w-event DP guarantee.

2.2 Privacy Definition

In this part, we provide the general definition of Differential Privacy (DP) in the context of data stream release. Then we introduce the definition of w-event DP and its semantic meaning.

Definition of Differential Privacy for Continuous Data. We present the definition of DP as follows:

Definition 1. *(ϵ-*Differential Privacy*) [17]. An algorithm $\mathcal{M}$ satisfies ϵ-differential privacy, where $\epsilon \geq 0$, if and only if for any neighboring streams $\mathcal{H}, \mathcal{H}'$, and any possible output set $O \subseteq Range(\mathcal{M})$, we have*

$$\Pr\left[\mathcal{M}(\mathcal{H}) \in O\right] \leq e^{\epsilon} \Pr\left[\mathcal{M}(\mathcal{H}') \in O\right]$$

Here, ϵ indicates *privacy budget*, which is a non-negative parameter that measures the privacy loss in the data. The smaller ϵ means that the outputs of $\mathcal{M}$ on two neighboring streams are more similar, and thus the provided privacy guarantee is stronger.

Intuitively, DP is guaranteed by adding carefully crafted noise to the statistical results. There are many existing mechanisms for sampling noise, such as Laplace mechanism, Gaussian mechanism, and Exponential mechanism [20]. Among these, the Laplace mechanism satisfies ϵ-DP protection, whereas the Gaussian mechanism achieves (ϵ, δ)-DP. The Exponential mechanism is commonly used to selection problems. This paper uses Laplace mechanism where the noise $\left\langle \text{Lap}\left(\Delta_f\right)\right\rangle^d$ are drawn from a Laplace distribution. The probability density function of Laplace distribution is

$$\Pr(x) = \frac{1}{2\lambda} e^{\frac{-|x|}{\lambda}} \tag{1}$$

where $\lambda = \Delta_f/\epsilon$ [17]. Δ_f is the l_1-norm global sensitivity of the statistical result, which is defined as $\Delta_f = \max_{D_i, D_i'} |f(D_i) - f(D_i')|$.

When processing a series of DP mechanisms and the statistical outcomes of these mechanisms are correlated, the privacy guarantee can be calculated using the privacy composition theory. We present the definition of sequential composition as follows:

Definition 2. *(*Sequential Composition*) [39]. For a sequence of k mechanisms $\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_k$ and $\mathcal{M}_i$ provides ϵ_i-DP, let $\mathcal{H}$ be a data stream, publishing $o = \{o_1, o_2, .., o_k\}$, where $o_1 = \mathcal{M}_1(\mathcal{H}), o_2 = \mathcal{M}_2(o_1, \mathcal{H}), ..., o_k = \mathcal{M}_k(o_1, ..., o_{k-1}, \mathcal{H})$, satisfies $(\sum_{i=1}^{k} \epsilon_i)$-DP.*

Definition of w-Event Differential Privacy. Events that occur at close timestamps in a data stream often exhibit correlations, which implies that the privacy protection provided by the event-level privacy may not satisfy the necessary requirements of dealing with privacy protection for scenarios involving continuous data publication. For instance, if a user's browser configuration remains constant for a week, reporting the same data across seven days, the privacy budget could expand from ϵ to 7ϵ due to the privacy composition theory.

w-event privacy is proposed to offer ϵ-DP protection for events occurring within any adjacent w timestamps. We present the formal definition of w-event privacy for neighboring data streams below,

Definition 3. *(w-Neighboring) [31]. Define H_i and H'_i as histograms of $\mathcal{H}$ and $\mathcal{H}'$ at the i^{th} timestamps. For a positive integer w, two streams $\mathcal{H}$ and $\mathcal{H}'$ are w-neighboring, if*

- *for each H_i, H'_i such that $H_i \neq H'_i$, it holds that H_i, H'_i are neighboring;*
- *for each $H_{i_1}, H_{i_2}, H'_{i_1}, H'_{i_2}$ with $i_1 < i_2$, $H_{i_1} \neq H'_{i_1}$ and $H_{i_2} \neq H'_{i_2}$, it holds that $i_2 - i_1 + 1 \leq w$.*

According to Definition 3, w-Neighboring requires that the histograms at the same timestamp of the two neighboring streams are adjacent or the same, and all the adjacent histograms fit in a window of up to w timestamps. In this paper, we follow the standard definition of DP to define neighboring histograms, meaning we consider neighboring histograms to differ by only one data entry. Therefore, the sensitivity of adding noise to each bin of the histogram is defined as $\Delta_f = 1$. Then, we show the definition of w-event DP in Definition 4.

Definition 4. *(w-Event ϵ-Differential Privacy) [31]. An algorithm $\mathcal{M}$ satisfies w-event ϵ-differential privacy, where $\epsilon \geq 0$, if and only if for any w-neighboring streams $\mathcal{H}, \mathcal{H}'$, and any possible output set $O \subseteq Range(\mathcal{M})$, we have*

$$\Pr[\mathcal{M}(\mathcal{H}) \in O] \leq e^{\epsilon} \Pr\left[\mathcal{M}(\mathcal{H}') \in O\right]$$

Semantic Meaning of w-event DP. w-event DP is a compromise between event-level DP (protecting only every single event) and user-level DP (shields all events associated with a specific user). It provides a privacy guarantee stronger than event-level DP and is considerably more practical than user-level DP (user-level DP requires additional constraints for releasing infinite data streams, i.e., limiting user contributions [15, 18] or allocating infinitesimal privacy budgets [26, 28]).

Intuitively, w-event DP changes the scope of protection from *a single event* as in event-level DP to *a sequence of events*, with a key *sliding window* constraint. The sliding window concept makes w-event DP not just an extension of event-level DP where the definition of an event becomes a sequence of w events, but actually a more stringent notion, i.e., any algorithm satisfying w-event ϵ-DP also satisfies (a sequence of w) event-level ϵ-DP but not vice versa. Next, we explain this point in detail.

Most streaming data publication applications, e.g., data collection/publication at Apple [1] and LinkedIn [43], allocate the total privacy budget for consecutive non-overlapping time windows (a sequence of events). Specifically, Apple allocates a new privacy budget every day and LinkedIn resets its privacy budget on the 1st day of each month. Suppose LinkedIn collects data n times each month, and the total privacy budget for each month is ϵ. If LinkedIn collects n times of data between January 16 and January 31, and n times between February 1 and February 15, even though each collected data is protected by event-level ϵ-DP, they do not satisfy w-event ϵ-DP since the privacy budget consumed within the window from January 16 to February 15 is 2ϵ.

2.3 Existing Solutions

To satisfy w-event DP, a naive way (we call it Uniform) is to evenly distribute the privacy budget across all timestamps within a window, enabling data release while ensuring that the total privacy budget consumed within each window does not exceed ϵ. However, this misses the opportunity to add less noise when data within a sliding window exhibits similar patterns. The other extreme is to publish once at the beginning of each sliding window (we call it Sample) and reuse this published data for all later timestamps throughout the window. This captures the information accurately at the initial stage, but this approach may not perform well if data changes rapidly for later timestamps within a window.

The central issue in w-event DP revolves around the strategic allocation of limited privacy budgets to different timestamps, i.e., when to publish (and the non-published timestamps will reuse the published ones) and how much privacy budget we allocate (while keeping the total privacy budget within each sliding window below ϵ). To address this challenge, several schemes are designed to enhance based on the naïve Sample method. In essence, these schemes can be categorized into two types based on the strategies they employ to control the sampling process.

Sampling Controlled by the Interval. This strategy revolves around adjusting the frequency of data releases based on specified intervals within the sliding window [24, 49, 50]. Subsequently, the question of *'How can we determine the intervals of data release?'* remains an essential challenge to address. Existing schemes FAST, RescueDP, and SecWeb propose to design a PID controller [32] to adjust the sampling interval based on the error of historical published results.

We emphasize that although these schemes adaptively adjust the sampling interval, they fall short in non-data-dependent data sampling and non-data-dependent privacy budget allocation. Specifically, they determine the worthiness of releasing current data based on the sampling interval, adjusted by historical releases, rather than the value of the current data itself. This limitation prevents the schemes from promptly adapting to changes in the data stream. Moreover, their budget allocation heavily relies on the pre-defined parameter values and current sampling intervals. They tend to reserve budgets for the remaining sampled timestamps within the current sliding window, and often neglect the specific trends of the data stream and the size of the sliding window. These often result in either an underutilization of the privacy budget within a window or the potential risk of an insufficient privacy budget allocation, leading to suboptimal accuracy of noisy releases.

Sampling Controlled by the Threshold. Unlike fixed-interval sampling, threshold-controlled sampling releases data when specific conditions or thresholds are met [31, 34]. These could be based on the magnitude of change in data, or other predefined criteria. BA/BD consumes half of the current remaining privacy budget each time. This leads to an exponential decrease in the privacy budget for timestamps as time progresses within a window. As a consequence, the earliest sampled timestamps consume a substantial portion of the privacy budget, causing greater noise in later released outcomes. The other strategy DSFT/DSAT is to additionally define a sampling count within a window and assign the privacy budget of each sampled timestamp evenly. However, they manually set the sampling count as an empirical parameter, which still poses a potential for unreasonable privacy allocation.

Based on the above analysis, we deduce that the threshold-controlled sampling schemes also encounter challenges related

Table 1: Summary of existing solutions based on strategy and whether their components are data dependent (✓yes, ✗no). 'Historical data' and 'Current data' refer to the results released at past timestamps and the new data at the current timestamp, respectively.

Solutions	Publication Strategy	Data Sampling		Budget Allocation
		Historical-data Dependent	Current-data Dependent	Historical/Current Data Dependent
Uniform [31]	'Uniform'	✗	✗	✗
AdaPub [52]	'Uniform'	✗	✗	✗
PeGaSuS [11]	'Uniform'	✗	✗	✗
RGP [41]	'Uniform'	✗	✗	✗
Sample [31]	'Sample'	✗	✗	✗
BA, BD [31]	'Sample'	✗	✓	✗
G-event [12]	'Sample'	✗	✓	✗
FAST [24]	'Sample'	✓	✗	✗
RescueDP [50]	'Sample'	✓	✗	✗
SecWeb [49]	'Sample'	✓	✗	✗
DSFT [34]	'Sample'	✗	✓	✗
DSAT [34]	'Sample'	✗	✓	✗
Our SPAS	'Sample'	✓	✓	✓

to non-data-dependent privacy budget allocation. Moreover, their sampling method solely considers the current data, lacking a prediction of stream trends based on historical releases.

We summarize all existing related schemes in Table 1. Notably, we include two schemes that are not proposed under the w-event DP: PeGaSuS, an event-level scheme that can be directly employed under w-event DP; and FAST, a user-level scheme that serves as the foundational design for some w-event strategies. All of these methods are included in a benchmark [46].

3 Our Solution

We have identified that none of the existing schemes achieve both data-dependent data sampling and data-dependent privacy budget allocation simultaneously. Consequently, they encounter a compromised accuracy, as they are unable to adapt the sampling process to changes in the data stream and allocate privacy budgets rationally.

To tackle this challenge, we design SPAS (Stream Publishing based on Adaptive SVT), which achieves both data-dependent sampling and data-dependent privacy budget allocation. SPAS predict the data publishing strategy across the entire sliding window, ensuring a well-planned allocation of privacy budget for both current and future releases, while maintaining the total consumption within a sliding window below ϵ. Furthermore, the publishing strategy also plays a crucial role in controlling the data sampling process to optimize the accuracy of the results.

3.1 Overview

Before delving into the algorithm's details, we begin by abstracting a workflow for SPAS. We show a flowchart of SPAS in Figure 1. SPAS features a central controller that encompasses three essential functions: predicting data publishing strategy, allocating privacy budgets, and sampling valuable data for release. We elaborate on these functions as follows.

- *Data Publishing Strategy Prediction.* The controller predicts the data publishing strategy for future timestamps, drawing on the previously published results. Specifically, we update the optimal sampling count C_i^* for data releases within an entire sliding window at the timestamp i, which plays a crucial role in budget allocation and data sampling procedures. The determination of C_i^* arises from the optimization problem rooted in an error function, which represents the sum of errors introduced by adding noise to sampled data and errors introduced by data not being sampled. This error function is mainly influenced by the number of data chosen for release within a sliding window.
- *Privacy Budget Allocation.* After determining the optimal sampling count C_i^*, this function investigates how to allocate the privacy budget accordingly. SPAS treats the sampled released data as equally significant and allocates the privacy budget to them evenly which is denoted as $\epsilon_{p_i} = \epsilon_p / C_i^*$. ϵ_p is the whole privacy budget used to release data within a sliding window.
- *Adaptive Sampling Control.* Adaptive sampling control is the core part of SPAS that determines whether data can be released at the current timestamp. This decision is based on two primary considerations: (a) whether the current data is worth publishing, and (b) whether the remaining privacy budget within the sliding window supports the privacy budget required by the current data publishing strategy, i.e., optimal sampling count. In this paper, we design a weighted Sparse Vector Technique (SVT) as the controller for adaptive sampling in SPAS.

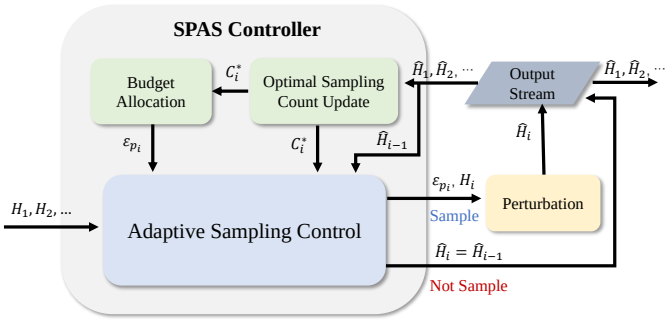

Figure 1: The workflow of SPAS. Initially, SPAS forecasts the data publishing strategy and determines the privacy budget for allocation. Subsequently, SPAS releases valuable data, drawing upon the preceding two steps.

Here, it is essential to declare which sliding window the term "current window" in the description of each function refers to since most timestamps are included in w sliding windows. In the process of predicting the data publishing strategy, we predict the optimal release count C_i^* for the sliding window starting from the current timestamp. In the process of adaptive sampling control, we calculate whether the remaining privacy budget within the sliding window ending at the current timestamp supports the privacy budget required for adding noise to the current data. The reason is that if the remaining privacy budget within this sliding window is sufficient, the available privacy budgets of the other $w-1$ sliding windows containing the current timestamp are also sufficient.

In the following sections, we will present the technical details of data publishing strategy prediction and adaptive sampling control.

3.2 Data Publishing Strategy Prediction

We predict the optimal sampling count as the data publishing strategy, which is crucial for sampling vital information and directing

the privacy budget allocation. It is imperative to highlight two critical distinctions between the optimal sampling count and the dynamic parameters, such as sampling interval and threshold values, used in existing related works. First, the prediction of the optimal sampling count is based on extensive historical data (covering multiple windows), rather than just the error from the most recent timestamps' results. Second, this optimal sampling count informs the data release strategy for the entire window, rather than concentrating exclusively on the present timestamp. It is noticed that our predictions are based on noisy results released in the past without repeated exposure to real streaming data to avoid additional privacy leakages. Then, we elaborate on how to calculate the optimal sampling count.

Estimate Optimal Sampling Count. We solve the optimal sampling count by minimizing an error function of the released results within a window. At each timestamp, we represent the error function as the total error of released results within the sliding window starting from this timestamp. Specifically, at the i^{th} timestamp, we estimate the error of released results $\{\hat{H}_i, \cdots, \hat{H}_{i+w-1}\}$. For convenience, we denote this segment of the released results as $\hat{\mathcal{H}}_w$, and its corresponding raw streaming data as $\mathcal{H}_w$. We assume that c_i timestamps in $\hat{\mathcal{H}}_w$ contain sampled data with noise added. We denote this subset of timestamps as $\mathcal{I}_s$, while the remaining $w - c_i$ timestamps can be represented as $\mathcal{I}_{\neg s}$. Then, we can formalize the error function for the variance of $\hat{\mathcal{H}}_w - \mathcal{H}_w$ as follows,

$$\text{Var}\left(\hat{\mathcal{H}}_w - \mathcal{H}_w\right) = \sum_{j \in \mathcal{I}_{\neg s}} \text{Var}\left(H_j - \hat{H}_{s_j}\right) + \sum_{j \in \mathcal{I}_s} \text{Var}\left(\text{Lap}\left(\frac{c_i \Delta_p}{\epsilon_p}\right)\right) \tag{2}$$

where $s_j \in \mathcal{I}_s$ and it is the maximum index of the timestamp smaller than j. For the timestamps $\mathcal{I}_{\neg s}$ where previous noisy results are directly released, the variance of the results comes from the difference between the true data and the previous noisy results. For the timestamps $\mathcal{I}_s$ where data is sampled to release, the variance of the released results comes from the Laplace noise $\text{Lap}\left(\frac{c_i \Delta_p}{\epsilon_p}\right)$, where Δ_p is the noise sensitivity and ϵ_p / c_i is the privacy budget used to add noise to each release.

In this paper, the adaptive sampling controller classifies the timestamps as belonging to either $\mathcal{I}_{\neg s}$ or $\mathcal{I}_s$ by determining whether the difference between the current data and the previously released result exceeds a threshold. We set the threshold to be $\frac{c_i \Delta_p}{\epsilon_p}$, which is the expected value of the absolute Laplace noise, as we consider that if the change in the current data is smaller than this threshold, its noisy result cannot reflect any valuable information. Based on Equation 2, we compute the optimal c_i in Theorem 1.

Theorem 1. *At the i^{th} timestamp, given ϵ_p as the total privacy budget for adding Laplace noise to sampled data within a window and Δ_p as the sensitivity of the noise, the optimal data sampling count C_i^* for the current window of length w is*

$$C_i^* = \left\lceil \frac{\epsilon_p}{6\Delta_p} \sqrt{3\text{Var}\left(\hat{E}^i_{dis}\right)} \right\rceil$$

where i and w are any positive integers smaller than the length of the data stream, and $\text{Var}\left(\hat{E}^i_{dis}\right)$ is the variance of distance between the noisy results in the current window.

Proof. The error comes from $\sum_{j \in \mathcal{I}_s} \text{Var}\left(\text{Lap}\left(\frac{c_i \Delta_p}{\epsilon_p}\right)\right)$ can be estimated from the probability density distribution of Laplace distribution shown in Equation 1. Since we cannot reuse the raw data of historically released results when estimating C_i^*, we cannot directly compute $\sum_{j \in \mathcal{I}_{\neg s}} \text{Var}\left(H_j - \hat{H}_{s_j}\right)$. We compute $\text{Var}\left(H_j - \hat{H}_{s_j}\right)$ as follows,

$$\begin{aligned}
&\text{Var}\left(H_j - \hat{H}_{s_j}\right) \\
&= \text{Var}\left(H_j - \hat{H}_{s_j} + \text{Lap}\left(\frac{c_i \Delta_p}{\epsilon_p}\right)\right) - \text{Var}\left(\text{Lap}\left(\frac{c_i \Delta_p}{\epsilon_p}\right)\right) \\
&= \text{Var}\left(\hat{H}_j - \hat{H}_{s_j}\right) - \text{Var}\left(\text{Lap}\left(\frac{c_i \Delta_p}{\epsilon_p}\right)\right) \\
&= \text{Var}\left(\hat{E}^i_{dis}\right) - \text{Var}\left(\text{Lap}\left(\frac{c_i \Delta_p}{\epsilon_p}\right)\right)
\end{aligned}$$

Here, we denote $\hat{H}_j - \hat{H}_{s_j}$ at the i^{th} timestamp as $\hat{E}^i_{dis}$.

The adaptive sampling controller introduces additional noise when using a variant of SVT to sample the data. Then we make the assumption that the variance of $x = \theta(w - c_i)$ streaming data at timestamps in $\mathcal{I}_{\neg s}$ is unbounded by the data sampling controller, whereas the variance of $w - c_i - x$ streaming data at timestamps in $\mathcal{I}_{\neg s}$ is bounded by the threshold of the data sampling controller, with θ representing the probability of the data sampling controller failing to sample the data whose change exceeds the threshold. We have

$$\begin{aligned}
&\text{Var}\left(\hat{\mathcal{H}}_w - \mathcal{H}_w\right) = x \cdot \text{Var}\left(H_i - \hat{H}_{s_i}\right) \\
&+ (w - c_i - x + c_i) \cdot \text{Var}\left(\text{Lap}\left(\frac{c_i \Delta_p}{\epsilon_p}\right)\right) \\
&= x \cdot \text{Var}\left(\hat{E}^i_{dis}\right) + (w - 2x) \cdot \text{Var}\left(\text{Lap}\left(\frac{c_i \Delta_p}{\epsilon_p}\right)\right) \\
&= \theta(w - c_i) \cdot \text{Var}\left(\hat{E}^i_{dis}\right) + ((1 - 2\theta)w + 2\theta c_i) \cdot \frac{2c_i^2 \Delta_p^2}{\epsilon_p^2}
\end{aligned}$$

Deriving the above formula, we can get that $\text{Var}\left(\hat{\mathcal{H}}_w - \mathcal{H}_w\right)$ is smallest when c_i satisfies

$$\frac{\sqrt{(1-2\theta)^2 w^2 + (3\theta^2 \text{Var}\left(\tilde{E}^i_{dis}\right)\epsilon^2)/\Delta^2} - (1-2\theta)w}{6\theta}$$

when $\theta = \frac{1}{2}$, we can simplify the optimal c_i as $\frac{\epsilon_p}{6\Delta_p}\sqrt{3\text{Var}\left(\hat{E}^i_{dis}\right)}$. □

Discuss Parameter θ. The parameter θ is influenced by multiple factors. We summarize following two main points.

(1) *Error caused by the adaptive sampling controller.* We use a variant of SVT as the adaptive sampling controller in this paper. The adaptive sampling controller has a probability of determining the data with changes exceeding the threshold as not for sampling, and this failure probability is contained within θ. Specifically, basic SVT is (α, β) accurate with $\alpha = 8(\log k + \log(2/\beta))/\epsilon$ (k is the number of data), which means that SVT does not sample the streaming data higher than $T - \alpha$ (T is the threshold) with a probability bounded

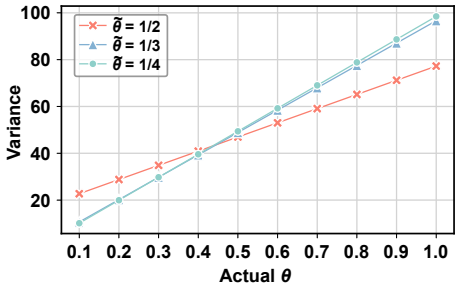

Figure 2: The variance of the results released in a window $\text{Var}\left(\hat{\mathcal{H}}_w - \mathcal{H}_w\right)$ with varying the actual θ, where the optimal sampling counts are calculated with $\tilde{\theta} = 1/2$, $\tilde{\theta} = 1/3$, and $\tilde{\theta} = 1/4$, respectively. We set $\epsilon_p = 1$, $\text{Var}\left(\hat{E}_{dis}\right) = 10$, and $w = 10$.

by $1-\beta$ [20]. (2) *Variation in the data stream distributions.* The adaptive sampling process follows a first-come-first-served approach. When the sampled count reaches the expected sampling count, sampling stops within the current window, which also results in some timestamps with data changes exceeding the threshold being classified into $\mathcal{I}_{\neg s}$. The proportion of such timestamps depends on the distribution of the data stream, that is the proportion of data with significant changes within the window. All of these factors complicate the theoretical analysis of the parameter θ.

We set three different theoretical values of $\tilde{\theta} = 1/2, 1/3, 1/4$ in Theorem 1 to get three different optimal sampling counts. We present in Figure 2 a comparison of $\text{Var}\left(\hat{\mathcal{H}}_w - \mathcal{H}_w\right)$ obtained using these three different optimal release counts as the true θ changes from 0.1 to 1. If $\tilde{\theta} > 1/2$, it implies that the accuracy of the adaptive sampling controller is significantly compromised. This results in a large theoretical variance of the results, so we exclude these lines in Figure 2. Given that $\text{Var}\left(\hat{\mathcal{H}}_w - \mathcal{H}_w\right)$ is minimally influenced by the actual value of θ when estimating the optimal sampling count with $\tilde{\theta} = 1/2$, we adopt $\tilde{\theta} = 1/2$ as the approximation of the true θ for both theoretical analysis and experimental evaluations.

Calculate $\text{Var}\left(\hat{E}^i_{dis}\right)$. In Theorem 1, the calculation of the optimal sampling count depends on a parameter $\text{Var}\left(\hat{E}^i_{dis}\right)$, which is the variance of the difference between the noisy results of the unsampled data and their adjacent released result within the current window. Apart from the i^{th} timestamp, the remaining $w-1$ data within the current window is still unknown, thus we cannot calculate the exact $\text{Var}\left(\hat{E}^i_{dis}\right)$.

To predict the optimal sampling count C^*_i, we estimate $\text{Var}\left(\tilde{E}^i_{dis}\right)$ as the approximate value of $\text{Var}\left(\hat{E}^i_{dis}\right)$ based on the historical released results at the timestamps in $\mathcal{I}_s$. We denote $\mathbf{o} = \{\hat{H}_1, \hat{H}_2, ..., \hat{H}_l\}$ as l noisy results released at sampled timestamps in $\mathcal{I}_s$ before the i^{th} timestamp. We calculate the variance of the distance between adjacent sampled noisy results $\tilde{E}^i_{dis}$ as,

$$\text{Var}\left(\tilde{E}^i_{dis}\right) = \frac{\sum_{j=2}^{l} |\hat{H}_j - \hat{H}_{j-1}|^2}{l-1} - \left(\sum_{j=2}^{l} \frac{|\hat{H}_j - \hat{H}_{j-1}|}{l-1}\right)^2$$

Algorithm 1 Weighted SVT

Input: A sliding window $\mathcal{H}_w = \{H_1, ..., H_w\}$, optimal sampling counts at each timestamp $C^* = \{C^*_1, ...C^*_w\}$, threshold T, privacy budget ϵ_s, sensitivity Δ_s of each streaming data.

$\epsilon_1 \leftarrow \epsilon_s/2$, $\epsilon_2 \leftarrow \epsilon_s/2$, $\rho \leftarrow \text{Lap}\left(\Delta_s/\epsilon_1\right)$, weight $\leftarrow 0$
for $H_i \in \mathcal{H}_w$ **do**
  $\nu_i \leftarrow \text{Lap}\left(2C^*_i\Delta_s/\epsilon_2\right)$
  if $dis_{H_i} + \nu_i > T + \rho$ **then**
    abort if weight $+ 1/C^*_i > 1$ ▷ dis_{H_i} *denotes the change in* H_i
    IsSampled $\leftarrow \top$
    weight $\leftarrow$ weight $+ 1/C^*_i$
  else
    IsSampled $\leftarrow \perp$
  end if
end for

Output: IsSampled

$\text{Var}\left(\hat{E}^i_{dis}\right)$ is the key parameter for calculating the optimal sampling count adapted to changes in the data stream. An overestimated $\text{Var}\left(\hat{E}^i_{dis}\right)$ can lead to an overestimated optimal sampling count, potentially resulting in incomplete utilization of the privacy budget within a sliding window. Conversely, an underestimated $\text{Var}\left(\hat{E}^i_{dis}\right)$ can lead to the budget within a sliding window being depleted prematurely. In this paper, we cannot theoretically estimate whether using $\text{Var}\left(\hat{E}^i_{dis}\right)$ to approximate $\text{Var}\left(\tilde{E}^i_{dis}\right)$ is unbiased due to the unknown nature of the upcoming data stream. We can only assume that the variation patterns of the data stream in several adjacent windows are similar. We will discuss in detail how to update $\text{Var}\left(\tilde{E}^i_{dis}\right)$ during the release process later.

3.3 Adaptive Data Sampling Control

This section focuses on how the optimal sampling count C^*_i guides the adaptive sampling controller in sampling the streaming data for release. This sampling process also needs to satisfy the w-event DP. We allocate a privacy budget of ϵ_s for data sampling in each sliding window. There are two objectives: one is to sample valuable streaming data for release, and the other is to ensure that the total privacy budget consumption within a window does not exceed ϵ_p.

To satisfy the first objective, we can use a technique called Sparse Vector Technique (SVT) [37] as the adaptive sampling controller. SVT regulates the continuous release of data exceeding a specified threshold T within a limited sampling count. Therefore, SVT allows us to focus on the most "interesting" or "significant" data without wasting the privacy budget on less important ones, making it well-suited for adaptive sampling control within the SPAS. In this paper, we can set the threshold T of SVT to be $\frac{C^*_i\Delta_p}{\epsilon_p}$, which is the expected value of absolute Laplace noise. We sample the current data and add noise to it for release only when the difference between the current data and the previously released result exceeds the threshold. In this way, we can ensure that the data changes in each released result are not obscured by the added noise at least.

For the second objective, it couldn't be satisfied using the existing SVT. The existing SVT maintains a predetermined maximum sampling count, denoted as C, and the sampling will stop when the number of sampled data reaches C. That is, the existing SVT

treats all sampled data as having equal weights, i.e., the weight of each sampled data is 1 and the total weight is C. However, in this paper, the privacy budgets ϵ_p/C_i^* allocated to each timestamp may be different since the optimal sampling count C_i^* calculated at each timestamp within a sliding window may vary. This means that each sampled data contributes a different weight to the total weight ϵ_p. To address this dilemma, we develop a weighted SVT, which can control the total privacy budget consumed by sampling data with dynamic privacy budgets. The detailed process of the weighted SVT within a sliding window is shown in Algorithm 1.

The form of Line 5 in Algorithm 1 is equivalent to checking if sampling the current data would result in the total privacy budget consumed within the window exceeding ϵ_p, since $\sum_{j\in[w]} 1/C_j^* > 1 \Leftrightarrow \sum_{j\in[w]} \epsilon_p/C_j^* > \epsilon_p$. The main difference between the existing SVT and Algorithm 1 lies in Line 5 and Line 7. The existing SVT checks weight + 1 > C (C is a constant) in Line 5 and computes weight ← weight + 1 in Line 7. Then, we prove that Algorithm 1 still provides w-event DP guarantee.

Theorem 2. *Weighted SVT satisfies w-event ϵ_s-DP.*

Proof. Consider any output vector set $\mathbf{o} = \langle o_1, o_2 ..., o_l \rangle$, where $o_i \in \{\perp, \top\}$ ("$\perp$" represents the data is not sampled, "$\top$" represents the data is sampled). Let $I_\perp = \{i : o_i = \perp\}$, $I_\top = \{i : o_i = \top\}$ be sets of "$\perp$" and "$\top$". Denote C_i^* as the optimal sampling count at the i^{th} output in $I_\top$. We have

$$\Pr[A(\mathcal{H}_w) = \mathbf{o}] = \int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in I_\perp} f_i(\mathcal{H}_w, z) \prod_{i\in I_\top} g_i(\mathcal{H}_w, z) dz$$

where $f_i(\mathcal{H}_w, z) = \Pr[dis_{H_i} + v_i < Ti + z]$, and $g_i(\mathcal{H}_w, z) = \Pr[dis_{H_i} + v_i \geq Ti + z]$. We also have $\Pr[\rho = z] \leq e^{\epsilon_1} \Pr[\rho = z + \Delta]$. Either the case that $\forall_i\ dis_{H_i} \geq dis'_{H_i}$ or $\forall_i\ dis_{H_i} \leq dis'_{H_i}$, we have

$$\begin{aligned} f_i(\mathcal{H}_w, z - \Delta) &= \Pr[dis_{H_i} + v_i < Ti + z - \Delta] \\ &\leq \Pr[dis'_{H_i} - \Delta + v_i < Ti + z - \Delta] \\ &= f_i(\mathcal{H}'_w, z) \end{aligned}$$

Since v_i is sampled from the distribution $\text{Lap}\left(\frac{2C_i^*\Delta}{\epsilon_2}\right)$, we have:

$$\begin{aligned} g_i(\mathcal{H}_w, z - \Delta) &= \Pr[dis_{H_i} + v_i \geq Ti + z - \Delta] \\ &\leq \Pr[dis'_{H_i} + \Delta + v_i \geq Ti + z - \Delta] \\ &\leq e^{\epsilon_2/C_i^*} \Pr[dis'_{H_i} + v_i \geq Ti + z] \\ &= e^{\epsilon_2/C_i^*} g_i(\mathcal{H}'_w, z) \end{aligned}$$

Therefore, we have:

$$\begin{aligned} &\frac{\Pr[A(\mathcal{H}_w) = \mathbf{o}]}{\Pr[A(\mathcal{H}'_w) = \mathbf{o}]} \\ &= \frac{\int_{-\infty}^{\infty} \Pr[\rho = z - \Delta] \prod_{i\in I_\perp} f_i(\mathcal{H}_w, z - \Delta) \prod_{i\in I_\top} g_i(\mathcal{H}_w, z - \Delta) dz}{\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in I_\perp} f_i(\mathcal{H}'_w, z) \prod_{i\in I_\top} g_i(\mathcal{H}'_w, z) dz} \\ &\leq \frac{\int_{-\infty}^{\infty} e^{\epsilon_1} \Pr[\rho = z] \prod_{i\in I_\perp} f_i(\mathcal{H}'_w, z) \prod_{i\in I_\top} e^{\epsilon_2/C_i^*} g_i(\mathcal{H}'_w, z) dz}{\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in I_\perp} f_i(\mathcal{H}'_w, z) \prod_{i\in I_\top} g_i(\mathcal{H}'_w, z) dz} \\ &= e^{\epsilon_1} e^{\sum_{i=1}^{|I_\top|} \frac{\epsilon_2}{C_i^*}} \leq e^{\epsilon} \end{aligned}$$

where $\sum_{i=1}^{|I_\top|} \frac{1}{C_i^*} \leq 1$, since the number of published $\top$ is always no greater than C_i^*. □

Algorithm 2 SPAS

Input: Data Stream $\mathcal{H} = \{H_1, ..., H_t\}$, initial sampling count C_{init}, privacy budget ϵ_s and ϵ_p, where $\epsilon_s + \epsilon_p = \epsilon$, dimension of each histogram d, the size of a sliding window w, the sensitivity $\Delta_s = \Delta_p = 1$.

Output: $\hat{\mathcal{H}} = \{\hat{H}_1, ..., \hat{H}_t\}$

1: $\epsilon_{s_1} \leftarrow \epsilon_s/2, \epsilon_{s_2} \leftarrow \epsilon_s/2, C_1^* \leftarrow C_{init}, \rho \leftarrow \text{Lap}\left(\Delta_s/\epsilon_{s_1}\right)$
2: **for** H_i **in** $\mathcal{H}$ **do**
3: $\quad dis_{H_i} \leftarrow \frac{1}{d} \sum_{j=1}^{d} |H_i[j] - \hat{H}_{i-1}[j]|$
4: $\quad$ **if** $dis_{H_i} + \text{Lap}\left(2C_i^*\Delta_s/\epsilon_{s_2}\right) > C_i^*\Delta_s/\epsilon_p + \rho$ **then**
5: $\quad\quad \epsilon_{p_i} \leftarrow \epsilon_p/C_i^*$ **abort if** $\sum_{j=t-w+1}^{i} \frac{\epsilon_{p_j}}{\epsilon_p} > 1$
6: $\quad\quad \hat{H}_i \leftarrow H_i + \langle \text{Lap}\left(\Delta_p/\epsilon_{p_i}\right) \rangle^d$
7: $\quad\quad$ Obtain $\text{Var}\left(\tilde{E}_{dis}^i\right)$
8: $\quad\quad C_{i+1}^* \leftarrow \epsilon_p \sqrt{3\text{Var}\left(\tilde{E}_{dis}^i\right)}/(6\Delta_p)$
9: $\quad$ **else**
10: $\quad\quad \epsilon_{p_i} \leftarrow 0, \hat{H}_i \leftarrow \hat{H}_{i-1}$
11: $\quad$ **end if**
12: **end for**
13: **return** $\hat{\mathcal{H}}$

Additionally, we need to mention that, as Rogers et al. [44] pointed out, historical-release-dependent privacy budget allocation can fail advanced DP composition theory [20]. The sequential composition adopted in this paper is not affected.

3.4 Algorithm Design of SPAS

After exploring the technical details of SPAS, this section elaborates on its whole process, as shown in Algorithm 2.

We divide the privacy budget ϵ for each sliding window into two parts: ϵ_s and ϵ_p. ϵ_s is used for sampling the data with weighted SVT, while ϵ_p is used to add Laplace noise to sampled data. In this paper, we default to setting $\epsilon_s = \frac{1}{4}\epsilon$ and $\epsilon_p = \frac{3}{4}\epsilon$. A portion of the data at the beginning of the data stream, i.e., the first sliding window, would be used for the warm-up stage to obtain the initial C_{init}. The operations during the warm-up stage still satisfy w-event DP and we will discuss the details further later on. In Algorithm 2, we present the process starting from after the warm-up stage, with C_{init} as an input value.

At the i^{th} timestamp, a streaming data H_i qualifies for release only if the difference between H_i and the previously released result $\hat{H}_{i-1}$ exceeds the threshold (Line 4). As mentioned earlier, we define the threshold as the expected value of the absolute Laplace noise to be added to the data, i.e., $T = C_i^*\Delta_s/\epsilon_p$. In essence, we consider the updated release worthy of release only if its modifications are significant enough not to be obscured by noise variations. Besides, weighted SVT needs to further determine whether there is enough remaining privacy budget in the sliding window culminating at the current timestamp to support the release (Line 5).

Each sampled release is allocated a privacy budget ϵ_p/C_i^*. We denote ϵ_{p_i} as the privacy budget consumed for releasing data at each timestamp. The Laplace noise is generated corresponding to the dimensions of the streaming data H_i (Line 6). Note that, as we defined in Definition 3, neighboring streaming data only differ by one data entry, which means the noise sensitivity of each bin in histogram H_i is 1. Therefore, we have $\Delta_s = \Delta_p = 1$. The optimal sampling count is updated after each release (Line 7-Line 8), as

detailed in Theorem 1. For timestamps that are not sampled, the released result from the previous timestamp would be directly used as the release result for the current timestamp (Line 10).

4 Implementation Details

In this section, we discuss some implementation details of SPAS.

Obtaining Initial Sampling Count C_{init}. In this part, we discuss how to obtain the initial sampling count C_{init} in the absence of historical publication during the warm-up stage.

During the warm-up stage, it's still important to ensure that the released results do not violate w-event DP. We employ a fixed sampling interval method [24] to publish data for the warm-up stage, and we typically use the data within the first sliding window for warming up where the sliding window length is not excessively small. We set the sampling interval as a constant m and allocate $\epsilon m/w$ privacy budget for each sampled data. Then, for the timestamps where data is not sampled, the publication result from the previous timestamp is directly used for release. Finally, we can calculate C_{init} according to Theorem 1 based on the released results during the warm-up stage.

If m is set too large, it may result in publishing an excessive number of unnecessary streaming data during the warm-up stage, leading to an underestimated C_{init}. This may cause SPAS to miss some valuable streaming data in subsequent several windows. Conversely, if m is set too small, it may lead to an overestimated C_{init}, causing the privacy budget in subsequent windows not to be fully utilized. However, under the assumption that we have no prior knowledge of the data stream, it's not possible to determine an optimal m. Fortunately, the impact of m can gradually diminish as the sliding window moves. Additionally, the optimization deployed can also mitigate the issue of inadequate utilization of the privacy budget in subsequent windows, which will be discussed later on.

Updating $\mathrm{Var}\left(\tilde{E}^i_{dis}\right)$ from Historical Released Results. In this part, we discuss how to update $\mathrm{Var}\left(\tilde{E}^i_{dis}\right)$ in the implementation.

A basic method is to update using all historical releasing results. When updating $\mathrm{Var}\left(\tilde{E}^i_{dis}\right)$ at the i^{th} timestamp, we calculate $(\mathrm{Var}\left(\tilde{E}^i_{dis}\right)\times l+|\hat{H}_i-\hat{H}_l|^2)/(l+1)$, where l is the number of sampled timestamps before the i^{th} timestamp and $\hat{H}_l$ is the last one released before the i^{th} timestamp. Another way is to update only using the noisy results released at the sampled timestamps of the several most recent windows. The $\mathrm{Var}\left(\tilde{E}^i_{dis}\right)$ updated by the former is relatively stable and less affected by outliers. In contrast, the latter is more sensitive to changes in the data stream.

Then, we provide another method using *Probability Decay* that is a compromise between the above two methods. We can group the historical results as $G_1, G_2, ..., G_{\lceil l/w \rceil}$ at intervals of w, and assign weight coefficients to the timestamps in each group according to the distance between the window where it is located and the current timestamp when updating $\mathrm{Var}\left(\tilde{E}^i_{dis}\right)$. Then we update $\mathrm{Var}\left(\tilde{E}^i_{dis}\right)$ as follows, where b can be set to 2.

$$\mathrm{Var}\left(\tilde{E}^i_{dis}\right) \simeq \frac{1}{l-1}\sum_{j=2}^{l}\frac{1}{b^{\lceil (i-j)/w \rceil}}|\hat{H}_i-\hat{H}_{i-1}|^2 \tag{3}$$

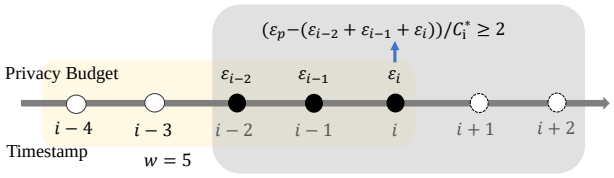

Figure 3: An example of maximizing the utilization of the privacy budget, where window size $w = 5$.

Maximizing the Utilization of the Privacy Budget. In the previous analysis, we identify that SPAS may face issues of either underutilization or unnecessary consumption of the privacy budget, indicating further optimization opportunities.

At each timestamp, if we find that the remaining privacy budget within the sliding window ending at the current timestamp is sufficient to support the publication of data from subsequent timestamps as the window slides, we directly publish them without judging by the adaptive data sampling controller. Specifically, as depicted in Figure 3, the window of length 5 encompasses three timestamps, spanning from $i-4$ to i. At the i^{th} timestamp, given the current optimal sampling count C_i^*, the remaining privacy budget is sufficient for releasing the results from the remaining timestamps within the new window, commencing from the initial sampling timestamp within the current window. In this case, the results from timestamps $i+1$ and $i+2$ can be directly released using privacy budget ϵ_p/C_i^* for each timestamp. Additionally, a portion of ϵ_{s_2} (Line 4 in Algorithm 2) does not need to be consumed for SVT judgment, thus the remaining part of ϵ_{s_2} can also be evenly distributed among the data directly released.

In the subsequent evaluation, we sets m to a constant value of 20 during the warm-up stage. The comparisons to other methods remain fair since the warm-up stage still satisfies the w-event DP. At each timestamp, the historically released results within the length of the previous two windows are used to calculate $\mathrm{Var}\left(\tilde{E}^i_{dis}\right)$. Additionally, the optimization for maximizing privacy budget consumption is applied.

5 Evaluation

5.1 Setup

Datasets. We evaluate all methods using six real-world datasets and three synthetic datasets. The six real-world datasets consist of three one-dimensional datasets and three multi-dimensional datasets. We summarize characters of all datasets in Table 2.

Processing for Real-world Datasets. We extended the length of all real-world datasets to approximately 4000 by concatenating the segments of the datasets. Longer data streams can allow the publishing strategies of the methods to update to the optimal state, which ensures that the comparison results of all methods are more reliable. For example, the group-based method PeGaSuS requires a sufficiently long group length to achieve the accuracy improvement; the accuracy of SPAS in the initial period of the stream is influenced by the parameter, i.e., c_{init}, manually set during the warm-up stage. In addition, we perform some other processing on each dataset to obtain a histogram data stream. The details are presented in our source code repository.

Figure 4: The comparison of all eight competitors on nine real-world and synthetic datasets with varying ϵ, where w is fixed to 120. The proposed method SPAS consistently demonstrates superior performance across all datasets.

Table 2: Datasets used for evaluation.

Dataset	Dimension	Domain Size	Timestamp
Flu Outpatient [3]	1	17,5650	1212
Flu Death [2]	1	1,626	1221
Unemployment [7]	1	612	609
Synthetic1	1	200	10,000
Synthetic2	1	1032	10,000
Synthetic3	10	10,000	10,000
State Flu [5]	55	22,904	628
TDrive [6]	64	2,627	2100
Retail [4]	100	693,099	143

Generation for Synthetic Datasets. We also generate three synthetic datastes, Synthetic1, Synthetic2, and Synthetic3, each with a length of 10, 000. For Synthetic1, we sample data from a uniformly random distribution ranging from 0 to 200. For Synthetic2, we sample data from a uniformly random distribution ranging from 0 to 1000 in the first half of the stream, and sample from a Gaussian distribution with a mean of 1000 and a variance of 10 in the second half of the data stream to simulate the distribution variation in real-world data streams. For Synthetic3, we sample the same two distributions as Synthetic2 with a data domain ranging from 0 to 10, 000, and it contains 10 dimensions.

Competitors. We compare SPAS with following competitors.

- Uniform **[31]**. It adds noise to the data at each timestamp with a uniform privacy budget ϵ/w.
- Sample **[31]**. It only samples the first timestamp in each window to add noise to the data. Each sampled timestamp is allocated with all the privacy budget, and the unsampled timestamps directly copy the results of the previous timestamp to release.
- BD **[31]**. It only releases data when there is a large change compared with the data at the previous timestamp, and it allocates the privacy budget in an exponentially decreasing manner.
- FAST **[24]**. It samples data according to a sampling interval and dynamically adjusts the sampling interval according to the past released results. Besides, it provides a user-level DP guarantee on the finite data stream, we modify it to provide w-event DP guarantee for comparison purposes.
- DSAT **[34]**. It controls the sampling process by using SVT, and only data whose change is greater than a dynamic threshold can be sampled for release. Since DSAT is an improved version of DSFT, we only compare with DSAT in this paper.
- AdaPub **[52]**. Similar to Uniform method. It also allocates the privacy budget evenly to each data for release, and it adds a smoothing operation to the noisy results.
- PeGaSuS **[11]**. It groups similar data at continuous timestamps using the clustering method and then smooths the noisy data within each group.

Metrics. We use Mean Relative Error (MRE) as the accuracy metric, measuring the difference between released results and the actual data streams. Specifically, for the released result $\hat{H}_i$ at the i^{th} timestamp, its MRE is defined as,

$$\text{MRE} = \frac{1}{n} \sum_{1 \le i \le n} |\frac{H_i - \hat{H}_i}{H_i}|. \tag{4}$$

where n is the total number of timestamps in the stream. For multi-dimensional $\hat{H}_i$, we calculate the MRE for each bin separately, and then take the average of all MRE.

We also evaluate with δ_{MRE} for a clearer MAE comparison across several methods, which is a special version of MRE introduced

Figure 5: The comparison of all eight competitors on nine real-world and synthetic datasets with varying w, where ϵ is fixed to 1. The proposed method SPAS **consistently demonstrates superior performance across all datasets.**

in [46]. It displays the relative deterioration in MRE for each method when compared to the optimal method's MRE. In this paper, we simply compute the δ_{MRE} as

$$\delta^m_{\text{MRE}} = \frac{\text{MRE}_m}{\min\{\text{MRE}_{m'} | m' \in M\}} \tag{5}$$

M represents the set of all methods, MRE_m refers to the MRE of method m.

Environment. All methods are implemented using Python 3.10.6 and the numpy 1.18.1 library. Experiments are carried out on servers running Ubuntu 22.04.1, equipped with E5-2620 v4 2.10GHz processors and 128 GB of memory.

5.2 Compare All Methods for Stream Release

To verify the effectiveness of SPAS, we assess the accuracy comparison of all 8 methods on six real-world datasets and three synthetic datasets with varying the key privacy parameters, i.e., privacy budget ϵ or window size w. We present δ_{MRE} in the form of heat maps, with the light color indicating higher accuracy, and a grid with a value of 1.0 indicating that it is the optimal result among all methods. The experiment results are presented in Figure 4 and Figure 5. Next, we will analyze the experimental results in detail.

Comparison of Privacy Budget ϵ. In Figure 4, we vary privacy budget ϵ from 0.1 to 0.9, and compare the δ_{MRE} of all methods on nine datasets. The size of each sliding window is fixed to 120. The first 5 rows of each heat map are the comparison results on one-dimensional datasets, and the last 4 rows are the results on multi-dimensional datasets. Figure 4 presents that every method except SPAS struggles to perform consistently well across all datasets. While all of them exhibit poor performance on specific datasets, it becomes challenging to determine the most suitable choice for handling unknown data streams in real-world situations.

We note that as ϵ increases, the weakness of sample-based methods relative to uniform-based methods becomes more evident for datasets where sample-based methods underperform. The underlying reason can be attributed to that the sampled timestamps fail to adequately cover the dynamic changes within the sliding window or to capture crucial data for release. As ϵ increases, the output from sample-based methods shows marginal enhancement, whereas the accuracy of results from uniform-based methods significantly improves, further highlighting the difference between two kinds of methods. Conversely, for datasets where uniform-based methods underperform, the disadvantage relative to sample-based methods decreases as ϵ increases. This outcome can be ascribed to the excessive release of sampled data, resulting in an inadequate allocation of the privacy budget for each sampled timestamp. As the privacy budget increases, the multitude of sampled timestamps start to present their strengths, narrowing the gap with sample-based methods.

Additionally, we observe that BD performs the poorest among the other baselines. The primary reason behind this trend is its lack of dependency on specific streaming data for privacy budget allocation, it tends to exhaust the majority of the privacy budget in the initial sampled timestamps. Consequently, in scenarios with small privacy budgets, i.e., $\epsilon < 1$, especially during data-intensive releases, the accuracy of BD may experience a significant decline.

Comparison of Window Size w. In Figure 5, we explore the effects of varying w from 80 to 240 and compare the δ_{MRE} of all methods across nine datasets, while maintaining a fixed total privacy budget of 1. Notably, SPAS consistently delivers robust performance across all datasets and for all values of w.

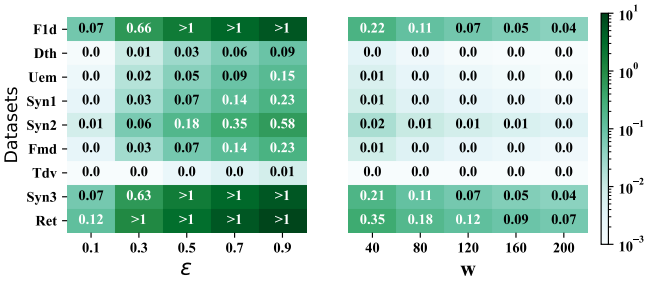

Figure 6: The comparison of characters across all datasets. Deeper colors represent higher variability in the streams.

The trends observed in the performance disparities between sample-based and uniform-based methods across different datasets remain consistent with the results presented in Figure 4. In datasets where the uniform-based methods exhibit superior performance, the limitations of sample-based methods are alleviated with increasing w. Although uniform-based methods capture a wider spectrum of updated streaming data and attain greater accuracy than sample-based methods, the decrease in the privacy budget dedicated to each timestamp as w grows results in increased noise variance for each sampled timestamp. This weakens the advantage of uniform-based methods compared to sample-based methods.

Conversely, for datasets where the sample-based methods demonstrate superior performance, the disadvantages of the uniform-based methods intensify with increasing w. This can be attributed to the tendency of uniform-based methods to allocate the privacy budget to an excessive number of unnecessary timestamps in datasets characterized by minimal fluctuations, resulting in suboptimal accuracy in the released results. As w increases, the privacy budget is distributed across an even greater number of unnecessary timestamps, exacerbating the inaccuracy of the released outcomes.

Analysis of All Datasets. The comparison results in Figure 4 and Figure 5 show clearly performance differences between sample-based methods and uniform-based methods on nine different datasets. In this section, we analyze the different characteristics of these datasets to further understand the variations in the performance of competitors and the rationale behind the calculation of the optimal publishing count for SPAS. The results are shown in Figure 6.

We compute the ratio of the sum of the distance between data at adjacent timestamps within any window of each data stream to the variance of Laplace noise introduced by releasing all w data points. A larger ratio indicates that adding noise to more sampled data points for releasing introduces smaller errors compared to publishing previous results, which also means that uniform-based methods would perform better in this case, and vice versa. The trend of color depth variation in the datasets in Figure 6 is completely consistent with Figure 4 and Figure 5, which further corroborates this point. SPAS predicts the optimal publishing counts based on historical release results for the situation depicted in Figure 6, thereby achieving high adaptability for various data streams.

5.3 Impact of Key Parameters on SPAS

The SPAS involves some important parameters that may affect its performance. We evaluate the impact of these parameters on the

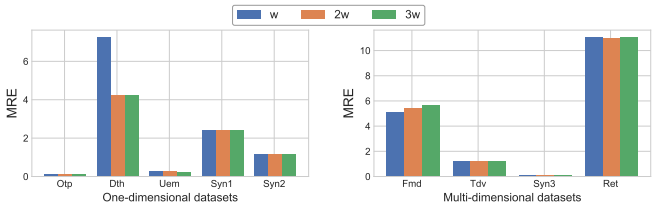

Figure 7: The accuracy comparison of SPAS when updating $\mathrm{Var}\left(\tilde{E}^i_{dis}\right)$ from different lengths of timestamps' released results, where the length are 1, 2, and 3 times the window size, respectively. We fix $\epsilon = 1$ and $w = 120$. There is no significant trend in accuracy across most datasets with varying lengths of timestamps.

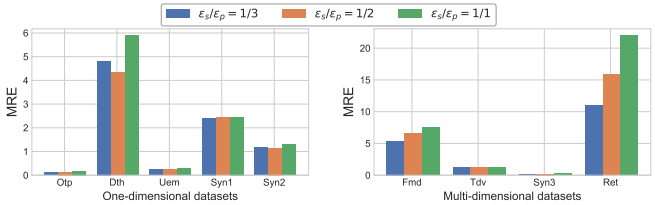

Figure 8: The accuracy comparison of SPAS when allocating different proportions of the privacy budget to ϵ_s and ϵ_p, where the ratios are 1/3, 1/2, and 1/1, respectively. We fix $\epsilon = 1$ and $w = 120$. Allocating more privacy budget to ϵ_p can achieve higher accuracy.

accuracy of the released results of SPAS across all experimental datasets. We fix $\epsilon = 1$ and $w = 120$ during the evaluation.

First, we compare the impact of using historically released results in different lengths of timestamps to update the optimal sampling count on the accuracy of the released results. Specifically, we use the released results at the timestamps within a length of 1, 2, and 3 window size before the current timestamp to estimate $\mathrm{Var}\left(\tilde{E}^i_{dis}\right)$. We observe that the MRE of the released results is not significantly affected. Only the results on the Flu Death dataset show a changing trend. As the length of the timestamps increases, the MRE of released results decreases.

Next, we compare the impact of the proportion of the privacy budget allocated to controlling the data sampling and data releasing on the accuracy of the released results. We allocate privacy budget of 1/3, 1/2, and 1/1 to ϵ_s and ϵ_p, respectively. Although the degree of impact reflected on each dataset is different, the overall trend is that it is better to allocate more privacy budget to ϵ_p. Two advantages can be brought by a large ϵ_p. One is that a large ϵ_p can introduce small noise to the sampled data. The other is that the threshold T can be also small, thus the error bound for unsampled data is also small. However, this may also increase the failure probability θ of weighted SVT, thus too small ϵ_s may also lead to a decrease in accuracy. We believe that $\epsilon_s/\epsilon_p = 1/3$ is a reasonable ratio for general data streams.

5.4 Compare All Methods for Range Query

We also compare the performance of the data streams released by eight competitors in responding to the range query tasks. We

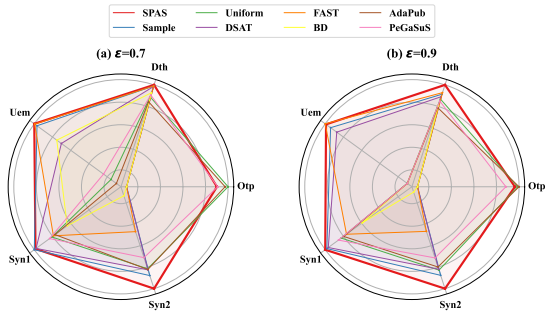

Figure 9: The comparison of all eight competitors for range query task with $\epsilon = 0.7$ and $\epsilon = 0.9$, where w is fixed to 120. SPAS performs the best on almost all datasets, except for a slight gap on the Flu Outpatient dataset.

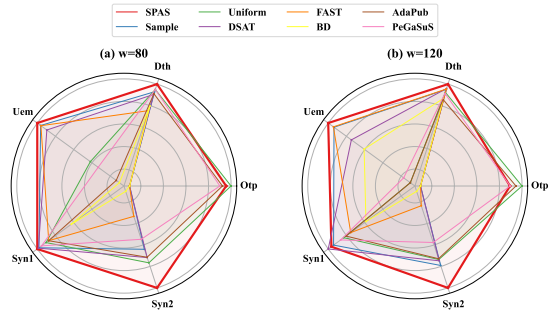

Figure 10: The comparison of all eight competitors for range query task with $w = 80$ and $w = 120$, where ϵ is fixed to 1. SPAS performs the best on almost all datasets, except for a slight gap on the Flu Outpatient dataset.

randomly generate 1,000 query requests, each with a range $[x, y)$ where both x and y are smaller than the upper bound of the data domain and $x < y$. Each query answer is the count of timestamps in the data stream that fall within the range. To highlight the differences in accuracy of the released results among different approaches, we still calculate δ_{MRE} after averaging the MRE of all query answers. The experimental results are shown in Figure 9 and Figure 10, respectively.

Each radar chart reflects the comparison results on five real-world and synthetic one-dimensional datasets. The outermost circle represents a $\delta_{MRE} = 1$, and adjacent circles differ by 2. The closer to the center, the higher the δ_{MRE}, indicating lower accuracy of the range query results. In Figure 9, we fix the window size $w = 120$ and display the comparison results for $\epsilon = 0.7$ and $\epsilon = 0.9$. In Figure 10, we fix $\epsilon = 1$ and show the comparison results for $w = 80$ and $w = 120$. We can observe that, except for a slight gap on the Flu Outpatient dataset, the range query accuracy of SPAS almost reaches the outermost circle for all datasets, while all other methods fail to achieve this. Especially on the Synthetic2 dataset, almost all other methods are located on the circle far from the outermost circle by more than one grid.

6 Related Work

Data Stream Publishing. Data stream publishing is a widely studied application in the field of data processing. Some works are dedicated to meeting the high volume requirements of data streams in limited memory by storing data in compressed data structures [35, 36, 38, 54, 55]. Some works focus on dealing with the velocity and volatility requirements of data streams, and many efficient data mining and analysis algorithms are designed [8, 9, 13, 27].

As privacy issues attract more public attention, some works begin to focus on publishing data streams under Differential Privacy (DP) protection. Some researches mainly focus on publishing data streams under the guarantee of event-level [14, 18, 23, 42, 51] and user-level privacy [15, 16, 25, 26, 48]. The former can protect any single data in streams and the latter can protect the entire data stream. To solve the problem that event-level privacy is weak for correlated streams and user-level cannot be used for infinity streams, w-event privacy is proposed to provide a guarantee with its strength between these two privacy definitions.

The definition of w-event privacy was first proposed by Kellaris et al. [31]. They mention that the privacy budget can be uniformly assigned to all data within a window or be assigned to a single data that is randomly sampled within a window. Besides, they also design two budget allocation schemes, Budget Distribution (BD) and Budget Absorption (BA), that optimize privacy allocation by reducing unnecessary data publish. Cheng et al. [12] propose G-event privacy based on BA to consider the difference between adjacent published data results from the dimension granularity. Then Li et al. [34] propose DSFT and DSAT to determine the sampling data by predefining a threshold. Wang et al. [49, 50] apply FAST, proposed in [24] with event-level privacy, to w-event privacy setting. There are also some works [11, 41, 41, 52] that use clustering methods to group streaming data and smooth the noisy data, thereby improving the accuracy of the published results. We recommend readers to refer to [46] to learn more w-event DP schemes.

Sparse Vector Technique (SVT). The Sparse Vector Technique (SVT) is employed to identify the first C queries whose answers are approximately larger than a predetermined threshold. Initially introduced by Dwork et al. [19], various specific algorithms were subsequently proposed [20, 29, 45]. Although some works attempted to propose the variants of SVT [10, 33, 47], they contained errors that violated differential privacy. Lyu et al. [37] categorized these incorrect variants and proposed an optimized SVT algorithm. In this paper, our design adopts the algorithm introduced in [37].

7 Conclusion

In this paper, we identified the fundamental issue with existing w-event DP solutions: their designs rely on experience or manual parameters that are detached from the specific data stream and lack planning for future timestamps, leading to suboptimal performance across diverse data streams. To address this, we proposed a novel scheme, SPAS, which integrates data-dependent processes for both data sampling and privacy budget allocation based on a predicted publishing strategy. Our comprehensive experimental evaluations demonstrated that SPAS can achieve high accuracy in released results across various data streams. Our future research aims to further enhance the data dependence of privacy budget allocation, incorporating predictive data publishing strategies based on historical data while considering the value of current data for privacy budget allocation.

References

[1] 2017. Apple Differential Privacy. https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf.

[2] 2023. Flu Death. https://gis.cdc.gov/grasp/fluview/mortality.html.

[3] 2023. Nation Flu Outpatient. https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html.

[4] 2023. Retail. https://www.kaggle.com/datasets/manjeetsingh/retaildataset.

[5] 2023. State Flu Outpatient. https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html.

[6] 2023. TDrive. https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/.

[7] 2023. Unemployment. https://fred.stlouisfed.org/series/LNU03000018.

[8] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. 2002. Counting Distinct Elements in a Data Stream. In *Randomization and Approximation Techniques, 6th International Workshop, RANDOM 2002, Cambridge, MA, USA, September 13-15, 2002, Proceedings (Lecture Notes in Computer Science, Vol. 2483)*, José D. P. Rolim and Salil P. Vadhan (Eds.). Springer, 1–10.

[9] Ran Ben-Basat, Gil Einziger, Roy Friedman, and Yaron Kassner. 2016. Heavy hitters in streams and sliding windows. In *35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016, San Francisco, CA, USA, April 10-14, 2016*. IEEE, 1–9.

[10] Rui Chen, Qian Xiao, Yu Zhang, and Jianliang Xu. 2015. Differentially Private High-Dimensional Data Publication via Sampling-Based Inference. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, Longbing Cao, Chengqi Zhang, Thorsten Joachims, Geoffrey I. Webb, Dragos D. Margineantu, and Graham Williams (Eds.). ACM, 129–138.

[11] Yan Chen, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2017. PeGaSus: Data-Adaptive Differentially Private Stream Processing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM, 1375–1388.

[12] Mian Cheng, Yipin Sun, Baokang Zhao, and Jinshu Su. 2016. An Event Grouping Approach for Infinite Stream with Differential Privacy. In *Advances in Services Computing - 10th Asia-Pacific Services Computing Conference, APSCC 2016, Zhangjiajie, China, November 16-18, 2016, Proceedings (Lecture Notes in Computer Science, Vol. 10065)*, Guojun Wang, Yanbo Han, and Gregorio Martínez Pérez (Eds.). 106–116.

[13] Graham Cormode and Marios Hadjieleftheriou. 2010. Methods for finding frequent items in data streams. *VLDB J.* 19, 1 (2010), 3–20.

[14] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting Telemetry Data Privately. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 3571–3580.

[15] Wei Dong, Qiyao Luo, and Ke Yi. 2023. Continual Observation under User-level Differential Privacy. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. IEEE, 2190–2207.

[16] Cynthia Dwork. 2010. Differential Privacy in New Settings. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, Moses Charikar (Ed.). SIAM, 174–183.

[17] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings (Lecture Notes in Computer Science, Vol. 3876)*, Shai Halevi and Tal Rabin (Eds.). Springer, 265–284.

[18] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. 2010. Differential privacy under continual observation. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, Leonard J. Schulman (Ed.). ACM, 715–724.

[19] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. 2009. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, Michael Mitzenmacher (Ed.). ACM, 381–390.

[20] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3-4 (2014), 211–407.

[21] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. 2010. Boosting and Differential Privacy. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*. IEEE Computer Society, 51–60.

[22] Michael Edward Edge, Pedro R. Falcone Sampaio, and Mohammed Choudhary. 2007. Towards a Proactive Fraud Management Framework for Financial Data Streams. In *Third IEEE International Symposium on Dependable, Autonomic and Secure Computing, DASC 2007, Columbia, MD, USA, September 25-26, 2007*. IEEE Computer Society, 55–64.

[23] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, Gail-Joon Ahn, Moti Yung, and Ninghui Li (Eds.). ACM, 1054–1067.

[24] Liyue Fan and Li Xiong. 2012. Real-time aggregate monitoring with differential privacy. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, Xue-wen Chen, Guy Lebanon, Haixun Wang, and Mohammed J. Zaki (Eds.). ACM, 2169–2173.

[25] Liyue Fan and Li Xiong. 2014. An Adaptive Approach to Real-Time Aggregate Monitoring With Differential Privacy. *IEEE Trans. Knowl. Data Eng.* 26, 9 (2014), 2094–2106.

[26] Farhad Farokhi. 2020. Temporally Discounted Differential Privacy for Evolving Datasets on an Infinite Horizon. In *11th ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS 2020, Sydney, Australia, April 21-25, 2020*. IEEE, 1–8.

[27] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. 2008. Graph Distances in the Data-Stream Model. *SIAM J. Comput.* 38, 5 (2008), 1709–1727.

[28] Shuya Feng, Meisam Mohammady, Han Wang, Xiaochen Li, Zhan Qin, and Yuan Hong. 2023. DPI: Ensuring Strict Differential Privacy for Infinite Data Streaming. *arXiv preprint arXiv:2312.04738* (2023).

[29] Moritz Hardt and Guy N. Rothblum. 2010. A Multiplicative Weights Mechanism for Privacy-Preserving Data Analysis. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*. IEEE Computer Society, 61–70.

[30] David J. Hill and Barbara S. Minsker. 2010. Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. *Environ. Model. Softw.* 25, 9 (2010), 1014–1022.

[31] Georgios Kellaris, Stavros Papadopoulos, Xiaokui Xiao, and Dimitris Papadias. 2014. Differentially Private Event Sequences over Infinite Streams. *Proc. VLDB Endow.* 7, 12 (2014), 1155–1166.

[32] Myke King. 2016. *Process control: a practical approach*. John Wiley & Sons.

[33] Jaewoo Lee and Christopher W. Clifton. 2014. Top-k frequent itemsets via differentially private FP-trees. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani (Eds.). ACM, 931–940.

[34] Haoran Li, Li Xiong, Xiaoqian Jiang, and Jinfei Liu. 2015. Differentially Private Histogram Publication for Dynamic Datasets: an Adaptive Sampling Approach. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, James Bailey, Alistair Moffat, Charu C. Aggarwal, Maarten de Rijke, Ravi Kumar, Vanessa Murdock, Timos K. Sellis, and Jeffrey Xu Yu (Eds.). ACM, 1001–1010.

[35] Jizhou Li, Zikun Li, Yifei Xu, Shiqi Jiang, Tong Yang, Bin Cui, Yafei Dai, and Gong Zhang. 2020. WavingSketch: An Unbiased and Generic Sketch for Finding Top-k Items in Data Streams. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 1574–1584.

[36] Xiaochen Li, Weiran Liu, Jian Lou, Yuan Hong, Lei Zhang, Zhan Qin, and Kui Ren. 2023. Local Differentially Private Heavy Hitter Detection in Data Streams with Bounded Memory. *CoRR* abs/2311.16062 (2023).

[37] Min Lyu, Dong Su, and Ninghui Li. 2017. Understanding the Sparse Vector Technique for Differential Privacy. *Proc. VLDB Endow.* 10, 6 (2017), 637–648.

[38] Gurmeet Singh Manku and Rajeev Motwani. 2002. Approximate Frequency Counts over Data Streams. In *Proceedings of 28th International Conference on Very Large Data Bases, VLDB 2002, Hong Kong, August 20-23, 2002*. Morgan Kaufmann, 346–357.

[39] Frank McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*, Ugur Çetintemel, Stanley B. Zdonik, Donald Kossmann, and Nesime Tatbul (Eds.). ACM, 19–30.

[40] Andres Molina-Markham, Prashant J. Shenoy, Kevin Fu, Emmanuel Cecchet, and David E. Irwin. 2010. Private memoirs of a smart meter. In *BuildSys'10, Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, Zurich, Switzerland, November 3-5, 2010*, Antonio G. Ruzzelli (Ed.). ACM, 61–66.

[41] Yiwen Nie, Liusheng Huang, Zongfeng Li, Shaowei Wang, Zhenhua Zhao, Wei Yang, and Xiaorong Lu. 2016. Geospatial Streams Publish with Differential Privacy. In *Collaborate Computing: Networking, Applications and Worksharing - 12th International Conference, CollaborateCom 2016, Beijing, China, November 10-11, 2016, Proceedings (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 201)*, Shangguang Wang and Ao Zhou (Eds.). Springer, 152–164.

[42] Victor Perrier, Hassan Jameel Asghar, and Dali Kaafar. 2019. Private Continual Release of Real-Valued Data Streams. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society.

[43] Ryan Rogers, Subbu Subramaniam, Sean Peng, David Durfee, Seunghyun Lee, Santosh Kumar Kancha, Shraddha Sahay, and Parvez Ahammad. 2021. LinkedIn's Audience Engagements API: A Privacy Preserving Data Analytics System at Scale. *J. Priv. Confidentiality* 11, 3 (2021).

[44] Ryan M. Rogers, Salil P. Vadhan, Aaron Roth, and Jonathan R. Ullman. 2016. Privacy Odometers and Filters: Pay-as-you-Go Composition. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (Eds.). 1921–1929.

[45] Aaron Roth and Tim Roughgarden. 2010. Interactive privacy via the median mechanism. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, Leonard J. Schulman (Ed.). ACM, 765–774.

[46] Christine Schäler, Thomas Hütter, and Martin Schäler. 2023. Benchmarking the Utility of w-event Differential Privacy Mechanisms - When Baselines Become Mighty Competitors. *Proc. VLDB Endow.* 16, 8 (2023), 1830–1842.

[47] Ben Stoddard, Yan Chen, and Ashwin Machanavajjhala. 2014. Differentially Private Algorithms for Empirical Machine Learning. *CoRR* abs/1411.5428 (2014).

[48] Hao Wang and Zhengquan Xu. 2017. CTS-DP: Publishing correlated time-series data via differential privacy. *Knowl. Based Syst.* 122 (2017), 167–179.

[49] Qian Wang, Xiao Lu, Yan Zhang, Zhibo Wang, Zhan Qin, and Kui Ren. 2016. SecWeb: Privacy-Preserving Web Browsing Monitoring with w-Event Differential Privacy. In *Security and Privacy in Communication Networks - 12th International Conference, SecureComm 2016, Guangzhou, China, October 10-12, 2016, Proceedings (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 198)*, Robert H. Deng, Jian Weng, Kui Ren, and Vinod Yegneswaran (Eds.). Springer, 454–474.

[50] Qian Wang, Yan Zhang, Xiao Lu, Zhibo Wang, Zhan Qin, and Kui Ren. 2016. RescueDP: Real-time spatio-temporal crowd-sourced data publishing with differential privacy. In *35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016, San Francisco, CA, USA, April 10-14, 2016*. IEEE, 1–9.

[51] Tianhao Wang, Joann Qiongna Chen, Zhikun Zhang, Dong Su, Yueqiang Cheng, Zhou Li, Ninghui Li, and Somesh Jha. 2021. Continuous Release of Data Streams under both Centralized and Local Differential Privacy. In *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi (Eds.). ACM, 1237–1253.

[52] Teng Wang, Xinyu Yang, Xuebin Ren, Jun Zhao, and Kwok-Yan Lam. 2019. Adaptive Differentially Private Data Stream Publishing in Spatio-temporal Monitoring of IoT. In *38th IEEE International Performance Computing and Communications Conference, IPCCC 2019, London, United Kingdom, October 29-31, 2019*. IEEE, 1–8.

[53] Shiming Yang, Konstantinos Kalpakis, and Alain Biem. 2014. Detecting Road Traffic Events by Coupling Multiple Timeseries With a Nonparametric Bayesian Method. *IEEE Trans. Intell. Transp. Syst.* 15, 5 (2014), 1936–1946.

[54] Tong Yang, Junzhi Gong, Haowei Zhang, Lei Zou, Lei Shi, and Xiaoming Li. 2018. HeavyGuardian: Separate and Guard Hot Items in Data Streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Yike Guo and Faisal Farooq (Eds.). ACM, 2584–2593.

[55] Yang Zhou, Tong Yang, Jie Jiang, Bin Cui, Minlan Yu, Xiaoming Li, and Steve Uhlig. 2018. Cold Filter: A Meta-Framework for Faster and More Accurate Stream Processing. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 741–756.