Dual Utilization of Perturbation for Stream Data Publication under Local Differential Privacy

Rong Du, Qingqing Ye, Yaxin Xiao, Liantong Yu, Yue Fu, Haibo Hu
Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University
roong.du@connect.polyu.hk, qqing.ye@polyu.edu.hk, yaxin.xiao@connect.polyu.hk,
liantong2001.yu@connect.polyu.hk, yuesandy.fu@connect.polyu.hk, haibo.hu@polyu.edu.hk

***Abstract*—Stream data from real-time distributed systems such as IoT, tele-health, and crowdsourcing has become an important data source. However, the collection and analysis of user-generated stream data raise privacy concerns due to the potential exposure of sensitive information. To address these concerns, local differential privacy (LDP) has emerged as a promising standard. Nevertheless, applying LDP to stream data presents significant challenges, as stream data often involves a large or even infinite number of values. Allocating a given privacy budget across these data points would introduce overwhelming LDP noise to the original stream data.**

Beyond existing approaches that merely use perturbed values for estimating statistics, our design leverages them for both perturbation and estimation. This dual utilization arises from a key observation: each user knows their own ground truth and perturbed values, enabling a precise computation of the deviation error caused by perturbation. By incorporating this deviation into the perturbation process of subsequent values, the previous noise can be calibrated. Following this insight, we introduce the Iterative Perturbation Parameterization (IPP) method, which utilizes current perturbed results to calibrate the subsequent perturbation process. To enhance the robustness of calibration and reduce sensitivity, two algorithms, namely Accumulated Perturbation Parameterization (APP) and Clipped Accumulated Perturbation Parameterization (CAPP) are further developed. We prove that these three algorithms satisfy w-event differential privacy while significantly improving utility. Experimental results demonstrate that our techniques outperform state-of-the-art LDP stream publishing solutions in terms of utility, while retaining the same privacy guarantee.

I. Introduction

In the era of big data, collecting and analyzing real-time data streams are essential in many distributed systems such as IoT, tele-health, and crowdsourcing, which continuously generate ample amount of data. Typical examples include real-time traffic updates for navigation systems such as Google Maps and Waze [12], and consumer sentiment analysis on peer review platforms like Yelp [36]. While these stream data offer substantial benefits for big data analysis and training AI models, they simultaneously bring forth privacy issues due to the potential exposure of sensitive personal information [5], [13].

Local Differential Privacy (LDP) [3], [6], [16], a cryptographic technique extensively deployed in various real-world contexts [4], [11], [23], [30], provides formal privacy guarantees by bounding the information leakage of individual users through a privacy parameter ϵ. However, its direct application to stream data faces fundamental limitations. In particular, user-level LDP [1] considers the worst-case scenario and partitions the privacy budget according to the composition theorem [19], where privacy budget division across timestamps leads to exponential utility degradation as accumulated noise overwhelms the original data. To address the challenge of limited privacy budget in stream data collection, two mainstream approaches have emerged in existing research, i.e., relaxing privacy definitions and reducing the amount of transmitted data.

For privacy relaxation, event-level LDP [1], [33] assigns independent privacy budgets ϵ to individual data points, which improves utility at the cost of weakened privacy guarantees across multiple timestamps. w-event LDP [17] provides a more rigorous privacy guarantee by constraining the budget allocation within sliding windows of length w. For reducing data transmission, methods proposed in [11], [34] allow users to report only at turning points, allocating more privacy budget to each reported point. However, reporting at turning points reveals temporal information, as the transmission time itself indicates when the original data changes. To decouple the relationship between turning points and time, Mao et al. [21] proposed PrivShape, which first employs symbolic aggregate approximation to convert numerical sequences into short strings, and then uses tree structures to identify frequent string patterns. However, due to the loss of detailed information, this method does not perform well in estimating means and distributions for range queries. Therefore, enhancing the utility while preserving user privacy remains a key challenge in LDP-enabled stream data analysis.

Departing from privacy budget allocation strategies, we propose a novel approach that leverages perturbation results of steam data in a dual manner. Beyond their conventional role in statistical analysis, these results can serve as valuable components for calibrating perturbation parameters. This dual-purpose approach stems from a key observation that each user knows both their own ground truth and perturbed values, which enables the deviation error to be computed precisely and in turn incorporated into subsequent values to calibrate the previous noise. Leveraging this auxiliary information, users can employ their locally perturbed values to adjust subsequent inputs for the perturbation mechanism $\mathcal{A}$, effectively mitigating errors from previous perturbations. We rigorously prove that this approach satisfies w-event LDP, as the adjust inputs process naturally dilutes individual value information.

To implement this idea, we first introduce the Iterative Perturbation Parameterization (IPP) algorithm as a strawman proposal, which incorporates the deviation from the previous perturbation to the current stream value as input for $\mathcal{A}$. Since the current perturbation is a cumulative effect of all previous data points rather than just the nearest one, we present the Accumulated Perturbation Parameterization (APP) algorithm to determine the deviations up to the current stream value. Through this process, the sensitivity increases as the range of the aggregated result expands. To mitigate this, we propose the Clipped Accumulated Perturbation Parameterization (CAPP), an optimized version of the APP that can reduce sensitivity while retaining most of the perturbation information. Additionally, we extend our perturbation parameterization scheme to time-slot sampling, which is particularly advantageous for mean statistics of subsequences while ensuring the accuracy of the published stream data.

Our contributions are summarized as follows:

- To the best of our knowledge, this is the first work that parameterizes input value based on perturbation for publishing stream data under LDP, which significantly enhances utility over existing state-of-the-art techniques.
- We define a new problem of subsequence data collection under LDP, for statistical analysis on subsequences influenced by LDP while retaining rigorous privacy guarantee, i.e., w-event LDP.
- We propose a time-slot sampling approach for subsequence mean statistics estimation over user sampling, which further improves the utility of subsequence mean estimation.

The remainder of this paper is organized as follows. In Section II, we introduce some preliminaries of LDP. In Section III, we formally define the problem and present a baseline approach to demonstrate our core ideas. In Section IV, we propose two optimized algorithms, followed by a sampling-based solution in Section V. In Section VI, we provide extensive experimental evaluations. Then we review related studies in Section VII, and conclude the paper in Section VIII.

II. Preliminaries

A. Local Differential Privacy

Local Differential Privacy (LDP) [6], [16] is a modern privacy-preserving standard that enables users to anonymize their data before publishing. It is defined as follows:

Definition 1. *A randomized algorithm $\mathcal{A}$ satisfies ϵ-local differential privacy (ϵ-LDP), if and only if for any two values x and x', and all possible outputs $y \subseteq$ Range($\mathcal{A}$), the following condition holds:*

$$P[\mathcal{A}(x) \in y] \leq e^{\epsilon} \cdot P[\mathcal{A}(x') \in y].$$

The intuition of LDP is the probabilistic nature that $\mathcal{A}$ maps any particular input to an output according to a distribution, controlled by the privacy budget ϵ. Therefore, anyone is unable to tell any individual's true answer from the observed output y with high confidence. Particularly, LDP lifts up the dependency on a trusted collector that is inherent in centralized differential privacy (DP) [8], [9], [22].

The following composition theorems of LDP provide a way to analyze the cumulative privacy loss when multiple LDP mechanisms are applied, either sequentially or in parallel.

Theorem 1. ***(Sequential Composition)*** *For any k mechanisms providing ϵ_i-local differential privacy for each, the sequence of all these mechanisms provides ϵ_{seq}-local differential privacy, where*

$$\epsilon_{seq} = \sum\nolimits_{i=1}^{k} \epsilon_i.$$

Theorem 2. ***(Parallel Composition)*** *Each of the k mechanisms provides ϵ_i-local differential privacy and operates on a disjoint subset of the entire dataset, then the union of these mechanisms provides ϵ_{par}-local differential privacy, where*

$$\epsilon_{par} = \max_i \{\epsilon_i\}.$$

These theorems are useful in theoretical analysis, as they allow the calculation of overall privacy loss in complex LDP scenarios and help in designing privacy-preserving algorithms by managing the privacy budget. The sequential composition theorem is particularly important for iterative algorithms or when a user's data is used multiple times, whereas the parallel composition theorem is applied when the dataset can be partitioned and separate analyses are run on each partition.

B. w-event Privacy

w-event privacy [17] is an LDP model tailed for data streams, which quantifies and limits privacy exposure within sliding windows containing w events. Specifically, this privacy model is defined on w-neighboring streams as follows.

Definition 2. ***(w-neighboring Streams)*** *Two streams $S = \{S_1, ..., S_t\}$ and $S' = \{S'_1, ..., S'_t\}$ of length t are w-neighboring streams if they have at most w consecutive different elements. Formally, for any $i, j \in \{1, \ldots, t\}$ and $i \leq j$, if $S_i \neq S'_i$ and $S_j \neq S'_j$, then $j - i + 1 \leq w$.*

Then we have the following definition of w-event privacy.

Definition 3. ***(w-event Privacy)*** *Let M be a mechanism that takes as input a stream of arbitrary size t and produces O as its output. M satisfies w-event ϵ-DP (or simply, w-event privacy) if for any two w-neighboring streams S_t and S'_t, the following inequality holds*

$$Pr[M(S) \in O] \leq e^{\epsilon} \cdot Pr[M(S') \in O].$$

A mechanism satisfying w-event privacy can provide ϵ-LDP guarantee in any sliding window of size w. In other words, for any mechanism with w-event privacy, ϵ can be viewed as the total available privacy budget in any sliding window of size w.

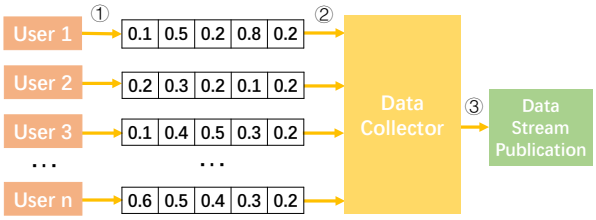


Fig. 1: Illustration of the data streams collection framework

C. Square Wave Mechanism

Square Wave (SW) [20] is a typical LDP mechanism for estimating numerical value distribution. Each user processes a numerical value in $[0, 1]$, and generates a sanitized value in $[-b, 1+b]$, where $b = \frac{\epsilon e^\epsilon - e^\epsilon + 1}{2e^\epsilon(e^\epsilon - \epsilon - 1)}$. Given an input value v, the randomized output can be expressed as following:

$$Pr[SW(v) = v'] = \begin{cases} p, & if |v - v'| \leq b, \\ q, & otherwise, \end{cases}$$

where $p = \frac{e^\epsilon}{2be^\epsilon + 1}$ and $q = \frac{1}{2be^\epsilon + 1}$. Upon receiving the perturbed data, the data collector aggregates the original distribution by using the Maximum Likelihood Estimation (MLE) [26], and reconstructs the distribution of original values.

III. Problem Definition and Baseline

In this section, we elaborate on our problem setting, and then introduce a baseline algorithm.

A. Data Collection Framework

Our problem setting involves a data collector and a group of distributed users, as shown in Figure 1. In Step ①, each user owns a continuous data stream. Since the data collector is untrusted, in Step ② the users utilize LDP mechanisms (e.g., SW [20]) to perturb their data and report the sanitized version. Upon receiving the perturbed data streams from the users, the data collector aggregates the inputs, with the goal of reconstructing an estimated data stream closely approximating the original one in Step ③. Finally, the data collector releases the aggregated values, e.g., mean or trends.

B. Problem Definition

The objective of the data collector is to estimate subsequences of users' data while ensuring w-event privacy (shown in Section II-B). Analyzing subsequences of data streams reveals localized meaningful patterns, which greatly aids data streams modeling and forecasting. To clarify, our discussion will primarily focus on a single data stream from one individual user, unless specified otherwise. Though this paper focuses on the single user scenario, the principles and algorithms discussed can be readily extended to accommodate multiple users.

The data collector aims to estimate a user's subsequence, denoted as $X_{(i,j)}$, which represents a continuous segment of the data stream spanning from the i-th time slot to the j-th time slot. Formally, it is defined as:

$$X_{(i,j)} = \{x_i, x_{i+1}, \ldots, x_j\}.$$

To protect their data, users will perturb the data using an LDP mechanism. In this paper, we employ the SW mechanism (shown in Section II-C), which is considered as state-of-the-art method for collecting numerical data. We suppose the subsequence perturbed by SW is:

$$X'_{(i,j)} = \{x'_i, x'_{i+1}, \ldots, x'_j\}.$$

The data collector will then reconstruct a subsequence $\hat{X}_{(i,j)}$ by aggregating the perturbed data streams collected over that period as

$$\hat{X}_{(i,j)} = \{\hat{x}_i, \hat{x}_{i+1}, \ldots, \hat{x}_j\}.$$

Directly releasing $\hat{X}_{(i,j)}$ is referred to as **stream data publication**. Aside from directly releasing the stream data, the data collector may also perform statistical analysis on $\hat{X}_{(i,j)}$ and publish the corresponding statistical results, such as calculating and publishing the mean or identifying trends. Specifically, the mean estimation for $\hat{X}_{(i,j)}$ is denoted as:

$$\hat{M}_{(i,j)} = \frac{\sum_{t=i}^{j} \hat{x}_t}{j - i + 1}.$$

C. Iterative Perturbation Parameterization (IPP)

In this section, we present a baseline algorithm named Iterative Perturbation Parameterization (IPP) for stream data publication, which iteratively adjusts input values based on perturbation results. The core idea of IPP is to integrate last perturbation deviation into the current value's perturbation process to mitigate errors. Specifically, let x_t denote the original value at the t-th time slot, and x'_t the corresponding perturbed value. Instead of directly perturbing the original value x_t, the user will calculate the deviation between x_{t-1} and x'_{t-1} and add it to x_t as the input value in order to partially correct the error caused by the perturbation in x_{t-1}. We give the details of the IPP algorithm as follows.

The procedure of IPP. Let x_t^I denote the input value at time slot t. As shown in Figure 2, our method is explained as follows. For the original value $x_1 = 0.01$ of the first time slot, since no data has been uploaded previously, the deviation is zero, and we have input value $x_1^I = x_1$. After perturbing by SW, we obtain $x'_1 = 0$. Since the user knows the original value and the perturbed one, the user can calculate the deviation $d_1 = x_1 - x'_1 = 0.01$, which will be corrected in the next time slot when perturbing x_2. The specific approach is to add this deviation to the second original value, and we obtain the input value $x_2^I = d_1 + x_2 = 0.16$. We then continue to perturb x_2^I and get a new perturbed value $x'_2 = 0.19$. From this, we obtain a new deviation $d_2 = x_2 - x'_2 = -0.04$, which is then added to x_3. IPP repeats this process until all values are collected.

It is important to note that when the original values are in the range of $[0, 1]$, adding the deviation may result in the range of the input value exceeding this range. To address this issue, we simply clip x_t^I to $[0, 1]$. Specifically, if $x_t^I < 0$, we set $x_t^I = 0$, and if $x_t^I > 1$, we set $x_t^I = 1$.

This algorithm achieves better utility, as proven by Lemma III.1. It is worth noting that the p, q, and b that appear in the subsequent proofs are parameters of SW mechanism, as illustrated in Section II-C.

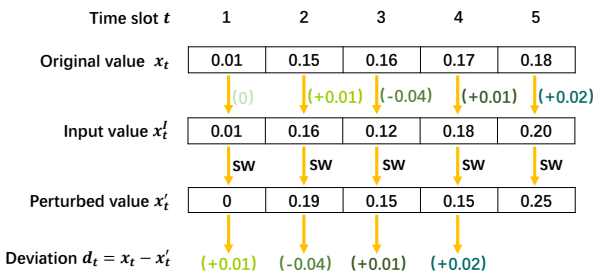


Fig. 2: The procedure of IPP

Lemma III.1. *Given data stream values* $X = \{x_1, x_2, \ldots, x_n\}$, *let* $MD(M) = \sum \frac{M(x_i)}{n} - \sum \frac{x_i}{n}$ *denote the mean deviation for algorithm* M. *We have* $MD(IPP) < MD(SW)$.

Proof. Let $d_1, \ldots, d_n$ denote the deviations obtained from the SW mechanism, where the mean deviation $MD(SW)$ is given by:

$$MD(SW) = \frac{SW(x_1) + \cdots + SW(x_n) - \sum x_i}{n}. \quad (1)$$

Let $d'_1, \ldots, d'_n$ denote the deviations obtained from the IPP algorithm, where the mean deviation for IPP, $MD(IPP)$, can be calculated as:

$$MD(IPP) = \frac{SW(x^I_1) + \cdots + SW(x^I_n) - \sum x_i}{n}. \quad (2)$$

Consider any two consecutive time slots t and $t-1$ in Equation 1. By expanding $SW(x_{t-1})$, we have:

$$T_{SW}(t-1, t) = \frac{x_{t-1} - d_{t-1} + SW(x_t) - (x_{t-1} + x_t)}{n}.$$

Consider any two consecutive time slots t and $t-1$ in Equation 2. By expanding $SW(x^I_{t-1})$, we have:

$$T_{IPP}(t-1, t) = \frac{x_{t-1} - d'_{t-1} + SW(x^I_t) - (x_{t-1} + x_t)}{n}$$
$$= \frac{x_{t-1} - d'_{t-1} + SW(x_t + d'_{t-1}) - (x_{t-1} + x_t)}{n}.$$

By design, the SW mechanism generates outputs that closely approximate its input x. Consequently, the term $x_{t-1} - d'_{t-1} + SW(x_t + d'_{t-1})$ yields results that better approximate the true value $x_{t-1} + x_t$, whereas the term $x_{t-1} - d_{t-1} + SW(x_t)$ introduces an additional error component d_{t-1}, resulting in larger deviations from the true value. Thus, we have

$$T_{IPP}(t-1, t) < T_{SW}(t-1, t). \quad (3)$$

Substituting Equation 3 into Equations 1 and 2, we obtain the following result:

$$MD(IPP) - MD(SW) =$$
$$\frac{1}{2}\left(\sum_{t=2}^{n} T_{IPP}(t-1, t) - \sum_{t=2}^{n} T_{SW}(t-1, t)\right) < 0,$$

which means IPP can always achieve lower mean deviation compared to directly perturbing stream data with the SW mechanism. □

IV. CAPP: Clipped Accumulated Perturbation Parameterization

Despite the benefits achieved by IPP in mitigating errors of the most recent perturbed value, it is limited to addressing a single perturbed value. A natural idea is extend our perturbation parameterization to encompass previous values, thereby enabling error correction from earlier perturbations. However, continuously accumulating deviations will lead to an unbounded increase in the input value range. Therefore, we need to constrain the range of input values reasonably. Based on these insights, we develop an advanced and effective algorithm called Clipped Accumulated Perturbation Parameterization (CAPP). To facilitate understanding, we first present Accumulated Perturbation Parameterization (APP) as an introductory solution, which considers all deviations caused by previously collected values and gives an effective post-processing method.

A. Accumulated Perturbation Parameterization (APP)

To account for accumulated perturbation errors in stream data collection, we maintain an accumulated deviation D of all perturbation-induced deviations from previously collected values. The key idea of APP is to adjust each new stream value by incorporating the deviation D as part of the input. The procedure of APP is presented in Algorithm 1.

Algorithm 1 Accumulated Perturbation Parameterization

Input: Privacy budget ϵ, window size w, original stream $\{x_i, \ldots, x_j\}$
Output: Collected data stream $\{x'_i, \ldots, x'_j\}$

1: $\epsilon_w = \epsilon / w$
2: Initialize accumulated deviation $D = 0$
3: **for** each time slot t **do**
4: $\quad x^I_t = truncate(x_t + D, [0, 1])$
5: $\quad x'_t = SW(x^I_t)$ with ϵ_w
6: $\quad$ Calculate deviation $d_t = x_t - x'_t$
7: $\quad D = D + d_t$
8: **end for**
9: **return** $\{x'_i, \ldots, x'_j\}$

The algorithm begins by calculating the privacy budget for each stream value, denoted by ϵ_w, to ensure the algorithm satisfies w-event privacy (line 1). It then sets an initial accumulated deviation D to zero since no value has been collected yet (line 2). We then compute x^I_t by adding D to the current value x_t, then clip x^I_t to [0,1] (line 4). The next step is the perturbation process, yielding the perturbed output x'_t (line 5). The algorithm then obtains the new deviation d_t (line 6) and adds it to D to update the accumulated deviation for future stream values (line 7). Finally, the data collector collects all the perturbed data (line 9).

Post-processing. After collecting the perturbed data, we proceed a smoothing step. The reason for including a smoothing step is that the perturbation process of the SW can introduce random positive or negative deviations. Smoothing

allows positive and negative deviations to counteract, thereby further reducing errors.

For smoothing, we employ the simplest method, the Simple Moving Average (SMA) [2]. The smoothing value $\text{SMA}_{x'_t}$ at time slot t, with a smoothing window size of $2k+1$, is as follows:

$$\text{SMA}_{x'_t} = \frac{1}{2k+1} \sum_{r=t-k}^{t+k} x'_r.$$

When dealing with boundary windows where the number of available values is less than $2k+1$, we simply average the available values. After smoothing, we can publish the stream data as follows:

$$\hat{X}_{(i,j)} = \{SMA_{x'_i}, \ldots, SMA_{x'_t}, \ldots, SMA_{x'_j}\}.$$

Smoothing has no impact on the mean of the results but can significantly improve the characterization of the data stream. The benefits of using SMA as post-processing of APP can be demonstrated by the following theoretical justifications:

Lemma IV.1. *Given a data stream $X = \{x_i, \ldots, x_t, \ldots, x_j\}$ and its APP-perturbed version $X' = \{x'_i, \ldots, x'_t, \ldots, x'_j\}$, define $Y = \{y_i, \ldots, y_t, \ldots, y_j\}$ as the smoothed stream obtained by applying simple moving average with window size s ($s > 1$) to X'. Then for any time point t, we have $Var(y_t) < Var(x'_t)$.*

Proof. Let $d_t = x_t - x'_t$ denote the deviation of x_t from its perturbed counterpart x'_t. The smoothed data point y_t, computed with a window size of $s = 2k+1$, is given by:

$$\begin{aligned} y_t &= \frac{x'_{t-k} + \ldots + x'_t + \ldots + x'_{t+k}}{2k+1} \\ &= \frac{x_{t-k} - d_{t-1} + \ldots + x_t - d_t + \ldots + x_{t+k} - d_{t+k}}{2k+1}. \end{aligned}$$

The expected value of y_t is formulated as:

$$\begin{aligned} E[y_t] = E\left[\frac{x_{t-k} \ldots + x_t + \ldots + x_{t+k}}{2k+1}\right] \\ - E\left[\frac{d_{t-k} \ldots + d_t + \ldots + d_{t+k}}{2k+1}\right]. \end{aligned}$$

Given that the deviations are bidirectional (both positive and negative) and exhibit compensatory behavior, where positive and negative deviations tend to counterbalance each other. Therefore, smoothing effectively mitigates the impact of perturbation noise for characterization. Next, we consider the variance of y_i. Since the noise is i.i.d., we have:

$$Var(y_t) = Var\left(\frac{x'_{t-k} + \ldots + x'_t + \ldots + x'_{t+k}}{2k+1}\right).$$

Due to the linearity of variance, we can express this as:

$$\begin{aligned} Var(y_t) &= \frac{1}{(2k+1)^2} Var(x'_{t-k} + \cdots + x'_t + \cdots + x'_{t+k}) \\ &= \frac{1}{(2k+1)^2} (Var(x'_{t-k}) + \ldots + Var(x'_t) + \ldots Var(x'_{t+k})). \end{aligned}$$

Because we do not know the specific results of the perturbed data, we consider the variance of all values to be the same as $Var(x'_t)$, from which we have:

$$Var(y_t) = \frac{(2k+1)Var(x'_t)}{(2k+1)^2} = \frac{Var(x'_t)}{2k+1}.$$

Thus, the variance of the smoothed value is smaller than that of the originally perturbed value. □

Theoretical analysis. The APP algorithm is designed to guarantee w-event privacy while achieving enhanced utility, as proven in Theorem 3, Lemma IV.2 and Lemma IV.3.

Theorem 3. *Given any two w-neighboring streams X and Y and an arbitrary output S from the APP algorithm, if each stream value is allocated a privacy budget of $\frac{\epsilon}{w}$, then the APP algorithm satisfies w-event privacy.*

Proof. Let's start with a straightforward scenario: a stream consisting of only two values, denoted as $X = \{x_1, x_2\}$ and $Y = \{y_1, y_2\}$, with a window size of $w = 2$. Let $S = \{s_1, s_2\}$ represent a possible output. Our APP algorithm begins to calculate the input values x_2^I and y_2^I only after receiving s_1. Applying Bayes' theorem [15], we arrive at the following equation:

$$\frac{P\{APP(x_1, x_2) = s_1, s_2\}}{P\{APP(y_1, y_2) = s_1, s_2\}} = \frac{P(s_1, s_2 | x_1, x_2 + x_1 - s_1)}{P(s_1, s_2 | y_1, y_2 + y_1 - s_1)}. \tag{4}$$

We will demonstrate the calculation of the joint distribution $P(s_1, s_2 | x_1,\ x_2 + x_1 - s_1)$. Using the chain rule of probability, we can break down this joint distribution:

$$\begin{aligned} &P(s_1, s_2 | x_1, x_2 + x_1 - s_1) \\ &= P(s_1 | x_1, x_2 + x_1 - s_1) P(s_2 | s_1, x_1, x_2 + x_1 - s_1). \end{aligned} \tag{5}$$

Firstly, $P(s_1 | x_1, x_2 + x_1 - s_1)$ simplifies to $P(s_1 | x_1)$ since s_1 is generated solely based on x_1 through the SW mechanism. For $P(s_2 | s_1, x_1, x_2 + x_1 - s_1)$, it simplifies to $P(s_2 | x_2 + x_1 - s_1)$, as s_1 and x_1 become redundant given $x_2 + x_1 - s_1$. Therefore, Equation 4 becomes:

$$\frac{P(s_1, s_2 | x_1, x_2 + x_1 - s_1)}{P(s_1, s_2 | y_1, y_2 + y_1 - s_1)} = \frac{P(s_1 | x_1) P(s_2 | x_2 + x_1 - s_1)}{P(s_1 | y_1) P(s_2 | y_2 + y_1 - s_1)}. \tag{6}$$

Since the APP algorithm primarily utilizes the SW mechanism, and each value is allocated a privacy budget of $\epsilon/2$ in this case, for any output u and any output o, the probability bounds can be expressed as $q \leq P(\text{APP}(u) = o) \leq p$. From this, we deduce:

$$\frac{P(s_1 | x_1)}{P(s_1 | y_1)} \leq \frac{p}{q} = e^{\frac{\epsilon}{2}}. \tag{7}$$

At the second time stamp, the user knows both x_1 and x_2. Moreover, s_1 is known to both the user and the data collector: the user knows it because the perturbation is executed locally, while the data collector receives s_1 during the first time step. Consequently, $x_2 + x_1 - s_1$ is a constant for user. Similar to Equation 7, we obtain:

$$\frac{P(s_2 | x_2 + x_1 - s_1)}{P(s_2 | y_2 + y_1 - s_1)} \leq \frac{p}{q} = e^{\frac{\epsilon}{2}}. \tag{8}$$

While the sensitivity of x_i in SW is 1, for $x_2 + x_1 - s_1$ it increases to 2. We apply clipping to restrict $x_2 + x_1 - s_1$ to [0,1], which reduces noise scale but without privacy leakage. Therefore, the clipping operation still maintains ϵ-differential privacy. Substituting equations (7) and (8) into equation (6), we obtain:

$$\frac{P(s_1, s_2 | x_1, x_2 + x_1 - s_1)}{P(s_1, s_2 | y_1, y_2 + y_1 - s_1)} \leq e^{\frac{\epsilon}{2}} e^{\frac{\epsilon}{2}} = e^{\epsilon}.$$

To demonstrate w-event privacy for the APP algorithm, we need to show that any perturbed subsequence of length w satisfies ϵ-differential privacy. Consider subsequences $\{x_1, \ldots, x_w\}$ from X and $\{y_1, \ldots, y_w\}$ from Y. With a corresponding outcome subsequence $S = \{s_1, \ldots, s_w\}$ and its reverse S^r, we derive:

$$\begin{aligned}&\frac{P\{APP(x_1, x_2, .., x_w) = S\}}{P\{APP(y_1, y_2, .., y_w) = S\}}\\ =&\frac{P\{APP(x_w, x_{w-1}, .., x_1) = S^r\}}{P\{APP(y_w, y_{w-1}, .., y_1) = S^r\}}\\ =&\frac{P(s_w|x_w^I)\ldots P(s_2|x_2^I)P(s_1|x_1)}{P(s_w|y_w^I)\ldots P(s_2|y_2^I)P(s_1|y_1)},\end{aligned} \tag{9}$$

where $x_t^I = x_t + D$. Since x_t and D are known constants for users, when we use x_t^I as the input of SW, we have $\frac{P(s_t|x_t^I)}{P(s_t|y_t^I)} < \frac{p}{q} = e^{\epsilon/w}$. Therefore, we conclude that:

$$\frac{P\{APP(x_1, x_2, .., x_w) = S\}}{P\{APP(y_1, y_2, .., y_w) = S\}} \leq (\frac{p}{q})^w \leq (e^{\epsilon/w})^w \leq e^{\epsilon}. \tag{10}$$

□

Lemma IV.2. *Given a subsequence $x_i, \ldots, x_t$ with corresponding deviations $b_i, \ldots, b_t$ and an accumulated deviation represented by D, let $ME(D)$ denote the deviation between the estimated mean and ground turth mean for the corresponding accumulated deviation D. We present the following inequality:*

$$ME(d_i, \ldots, d_t) < ME(d_{i+1}, \ldots, d_t) < \cdots < ME(d_t).$$

Proof. We prove this result inductively by analyzing streams of increasing length. Starting with a basic case of two consecutive values (x_{t-1} and x_t), we extend the analysis to three values, and finally generalize to an arbitrary stream length of $t - i + 1$ values. We consider the case when the x_t^I is not perturbed and the estimated mean is then:

$$\frac{x'_{t-1} + x_t^I}{2} = \frac{x'_{t-1} + x_t + x_{t-1} - x'_{t-1}}{2} = \frac{x_{t-1} + x_t}{2},$$

which is the true mean of $\{x_{t-1}, x_t\}$. Then we introduce x_{t-2}, we consider the case when the x_t^I is not perturbed similarly. The estimated mean is then:

$$\begin{aligned}&\frac{x'_{t-2} + x'_{t-1} + x_t^I}{3}\\ =&\frac{x'_{t-2} + x'_{t-1} + x_t + x_{t-1} - x'_{t-1} + x_{t-2} - x'_{t-2}}{3}\\ =&\frac{x_{t-2} + x_{t-1} + x_t}{3}.\end{aligned}$$

Similar to the case $\{x_{t-2}, x_{t-1}, x_t\}$, we can generalize this to t values: when the t-th input value is not perturbed, the estimated mean is:

$$\frac{x_i + x_2 + \cdots + x_t}{t - i + 1},$$

which is the same as the ground truth. Assume the perturbation result for x_t^I is R. The range of R is $[-b, 1 + b]$. As ϵ approaches 0, b approaches $\frac{1}{2}$. Therefore, the maximum range of R is $[-\frac{1}{2}, 1 + \frac{1}{2}]$. The final estimated mean deviation depends only on the perturbation R of the previous item. And we have:

$$ME(d_r, \ldots, d_t) = \frac{R}{t - r + 1}.$$

When $t - r \leq 1$, the influence of R on the numerator is less than that on the denominator, and we can conclude that the bias error increases as r becomes larger. □

Lemma IV.3. *Let $\{x_1, x_2, \ldots, x_n\}$ be the true time series, $\{x'_1, x'_2, \ldots, x'_n\}$ be the directly perturbed time series without APP, and $\{y_1, y_2, \ldots, y_n\}$ be the time series using APP and smoothing. Define the cosine similarities between the true time series and the two perturbed time series as $\cos(\theta\text{Direct})$ and $\cos(\theta\text{APP})$. Then, the following inequality holds:*

$$\mathbb{E}[\cos(\theta\text{APP})] > \mathbb{E}[\cos(\theta_{\text{Direct}})]$$

Proof. We demonstrate that APP combined with smoothing achieves higher cosine similarity with the original time series than direct perturbation. We define the cosine similarities as:

$$\cos(\theta_{\text{Direct}}) = \frac{\langle x, x' \rangle}{|x||x'|} = \frac{\sum_{i=1}^n x_i x' i}{\sqrt{\sum i = 1^n x_i^2}\sqrt{\sum_{i=1}^n (x' i)^2}}$$

$$\cos(\theta\text{APP}) = \frac{\langle x, y \rangle}{|x||y|} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2}\sqrt{\sum_{i=1}^n y_i^2}}$$

where $x'_i = x_i + \beta_i$ with β_i being random noise, and y_i is the APP-perturbed and smoothed time series. For direct perturbation, β_i is independent noise with $\mathbb{E}[\beta_i] = 0$ and $\text{Var}(\beta_i) = \sigma\beta^2$. This affects the cosine similarity in two ways. The numerator contains a noise term:

$$\langle x, x' \rangle = \sum_{i=1}^n x_i(x_i + \beta_i) = |x|^2 + \sum_{i=1}^n x_i \beta_i$$

While $\mathbb{E}[\sum_{i=1}^n x_i \beta_i] = 0$, the variance of this term is:

$$\text{Var}\left(\sum_{i=1}^n x_i \beta_i\right) = \sum_{i=1}^n x_i^2 \text{Var}(\beta_i) = \sigma_\beta^2 |x|^2$$

The denominator is inflated by noise, with expected squared norm:

$$\begin{aligned}\mathbb{E}[|x'|^2] &= \mathbb{E}\left[\sum_{i=1}^n (x_i + \beta_i)^2\right]\\ &= \sum_{i=1}^n (x_i^2 + 2x_i \mathbb{E}[\beta_i] + \mathbb{E}[\beta_i^2]) = |x|^2 + n\sigma_\beta^2\end{aligned}$$

In contrast, APP leverages prior knowledge to generate adaptive noise η_i with lower variance $\text{Var}(\eta_i) = \sigma_\eta^2 < \sigma_\beta^2$ and adaptively generated to align better with the original signal structure. The APP-perturbed and smoothed time series is defined as:

$$y_i = \frac{1}{s} \sum_{j=i-\lfloor s/2 \rfloor}^{i+\lfloor s/2 \rfloor} (x_j + \eta_j)$$

where s is the smoothing window size. The smoothing operation further reduces noise variance. Assuming independent noise, the variance of smoothed noise becomes:

$$\text{Var}\left(\frac{1}{s} \sum_{j=i-\lfloor s/2 \rfloor}^{i+\lfloor s/2 \rfloor} \eta_j\right) = \frac{1}{s^2} \sum_{j=i-\lfloor s/2 \rfloor}^{i+\lfloor s/2 \rfloor} \text{Var}(\eta_j) = \frac{\sigma_\eta^2}{s}$$

This dual advantage affects the cosine similarity: the numerator's noise component has reduced variance

$\text{Var}\left(\langle x, y\rangle - |x|^2\right) < \text{Var}\left(\langle x, x'\rangle - |x|^2\right)$, and the denominator experiences less inflation:

$$\mathbb{E}[|y|^2] \approx |x|^2 + \frac{n\sigma_\eta^2}{s}$$

Comparing the simplified expressions for expected cosine similarities:

$$\mathbb{E}[\cos(\theta_{\text{Direct}})] \approx \frac{1}{\sqrt{1 + \frac{n\sigma_\beta^2}{|x|^2}}}$$

versus

$$\mathbb{E}[\cos(\theta_{\text{APP}})] \approx \frac{1}{\sqrt{1 + \frac{n\sigma_\eta^2}{s|x|^2}}}$$

Since $\sigma_\eta^2 < \sigma_\beta^2$ and $s > 1$, we definitively have $\frac{n\sigma_\eta^2}{s|x|^2} < \frac{n\sigma_\beta^2}{|x|^2}$. Therefore $\mathbb{E}[\cos(\theta_{\text{APP}})] > \mathbb{E}[\cos(\theta_{\text{Direct}})]$. □

B. Clipped Accumulated Perturbation Parameterization (CAPP)

In the IPP and APP algorithms, input values are directly clipped to [0,1] before applying SW perturbation, which is a simplistic approach that does not consider the privacy budget or the impact of accumulated deviation on utility. In CAPP, we first clip the accumulated values to a range $[l, u]$, then normalize them to $[0, 1]$ for SW perturbation, and finally denormalize back to $[l, u]$. This approach provides more flexibility in handling accumulated deviations while maintaining privacy guarantees. The main content of this subsection introduces the CAPP algorithm and demonstrates how to determine optimal $[l, u]$ ranges to achieve better utility.

Algorithm 2 Clipped Accumulated Perturbation Parameterization

Input: Privacy budget ϵ, windows size w
Output: $\hat{X}_{(i,j)} = \{\hat{x}_i, \ldots, \hat{x}_t, \ldots, \hat{x}_j\}$

1: Determine l and u;
2: $\epsilon_w = \epsilon/w$;
3: Initialize accumulated deviation $D = 0$
4: **for** each time slot t **do**
5: Update input value: $x_t^I = x_t + D$
6: Clipping:

$$x_t^I = \begin{cases} l, & \text{if } x_t^I < l, \\ u, & \text{if } x_t^I > u. \end{cases}$$

7: Normalization: $x_t^I = \frac{x_t^I - l}{u - l}$
8: Perturbation: $x_t' = SW(x_t^I)$
9: Denormalization: $x_t' = x_t'(u - l) + l$
10: Calculate deviation: $d_t = x_t - x_t'$
11: $D = D + d_t$
12: **end for**
13: $\hat{X}_{(i,j)} = SMA(X'_{(i,j)})$;
14: **return** $\hat{X}_{(i,j)} = \{\hat{x}_i, \ldots, \hat{x}_t, \ldots, \hat{x}_j\}$

The procedure of CAPP. The Algorithm 2 initiates by determining the lower bound l and upper bound u for input values (line 1). The privacy budget per time slot ϵ_w is calculated (line 2), and an initial accumulated deviation D is set to zero (line 3). For each x_t within the stream, the algorithm first retrieves an input value based on D. It then ensures x_t^I remains within the $[l, u]$ bounds by clipping: if x_t^I falls below the lower bound l, it is raised to l, and if it exceeds the upper bound u, it is reduced to u (line 6). Next, the algorithm normalizes the clipped input value to the range [0,1]. This normalization satisfies the input range requirement of the SW algorithm (lines 7-8). Following this, the algorithm employs the SW to perturb x_t^I, and the perturbed value x_t' is then denormalized to the original range $[l, u]$ (line 9). It computes the deviation d_t (line 10) and the accumulated deviation D (line 11). Finally, the algorithm executes a smoothing procedure on the perturbed values and returns the estimated stream data (lines 13-14).

According to Theorem 4, CAPP satisfies ϵ-LDP while applying a clipping operation to the input values. This clipping operation reduces the noise scale without compromising privacy guarantees.

Theorem 4. *Given any two w-neighboring streams X and Y and an arbitrary output S from the CAPP algorithm with any clip range $[l, u]$, if each stream value is allocated a privacy budget of $\frac{\epsilon}{w}$, then the CAPP algorithm satisfies w-event privacy.*

Proof. For Equation 8, although $x_2 + x_1 - s_1$ is clipped to $[l, u]$, the normalization transformation ($x_{\text{norm}} = \frac{x-l}{u-l}$), which ensures the input is always restricted to $[0, 1]$, preserves the privacy guarantees. Since both clipping and normalization are deterministic operations, they neither introduce additional randomness nor alter the privacy budget of the original mechanism. Therefore, the privacy guarantee remains:

$$\frac{P(s_2|x_2 + x_1 - s_1)}{P(s_2|y_2 + y_1 - s_1)} \leq e^{\epsilon/2}.$$

The remaining proof follows the same steps as in Theorem 3.

□

The choice of l and u. The Algorithm 2 involves choosing l and u, which affect the utility of our method through different types of errors. As more stream values are collected, the range of x_t^I expands. Clipping x_t^I into a fixed range $[l, u]$ presents a trade-off: while a wider range preserves extreme values, it leads to higher sensitivity, requiring larger noise introduction and thus increasing sensitivity error e_s; conversely, a narrow range reduces data sensitivity and requires less noise for the same privacy guarantee, but excessive narrowing discards some information, introducing significant discarding error e_d.

The specific processes for calculating e_s and e_d are given as follows. We define an error function $T(e_s, e_d)$ to determine the clipping bound:

$$T(e_s, e_d) = e_s - e_d. \tag{11}$$

The calculation of $[l, u]$ is based on $T(e_s, e_d)$:

$$\begin{cases} l = 0 - T(e_s, e_d), \\ u = 1 + T(e_s, e_d). \end{cases}$$

Sensitivity error e_s. This error quantifies the deviation introduced by clipping. The calculation is defined as:

$$e_s = e^{x - E(SW(x))} - 1,$$

where $x - E(SW(x))$ measures the deviation between the original and expected perturbed values. Using $e^{x-E(SW(x))} - 1$ instead of simply $x - E(SW(x))$ to measure error has two advantages. First, it ensures e_s approaches 0 for large ϵ, where sensitivity reduction becomes unnecessary. Second, the exponential function mapping in $e^{x-E(SW(x))}-1$ can amplify even small differences between the original value and perturbed value. Given the unknown distribution of original data, we consider the worst-case scenario where $x = 1$.

Discarding error e_d. To characterize the discarding error e_d, we first establish the probability density function $D_x = x - SW(x)$ to analyze the deviation distribution. Based on this, we define e_d using the standard deviation of D_x. Formally:

$$e_d = \sqrt{Var(D_x)}.$$

Smaller ϵ leads to larger $Var(D_x)$, indicating more significant deviations, while clipping within a narrow range would result in excessive information loss. Next, we give the calculation process of $Var(D_x)$. First, the probability density function of $G(D_x)$ is:

$$G(D_x) = \begin{cases} q, & \text{if} \quad D_x \in [-1 - b + x, -b], \\ p & \text{if} \quad D_x \in (-b, b), \\ q & \text{if} \quad D_x \in [b, b + x]. \end{cases}$$

The expectation and variance of this distribution can be calculated, with the results as follows. The expected value of D_x is given by:

$$E(D_x) = q((1 + 2b)x - (b + \frac{1}{2})).$$

The expectation of $G(D_x^2)$ is:

$$E(D_x^2) = q\frac{-3b^2+6bx^2-6bx+3b+3x^2-3x+1}{3} + \frac{2pb^3}{3}.$$

For e_d computation, we similarly consider the worst-case scenario with $x = 1$ due to the unknown ground-truth. Then, the variance of D_x is:

$$\begin{aligned} Var(D_x) &= E(D_x^2) - (E(D_x))^2 \\ &= \frac{2b^3p}{3} - b^2q^2 + b^2q - bq^2 + bq - \frac{q^2}{4} + \frac{q}{3}. \end{aligned}$$

C. Discussion on Generalizability

Crowd-level statistics. We extend our analysis from individual-level to crowd-level statistics by leveraging the collective information from independently and identically distributed (i.i.d.) users. Specifically, we first estimate individual statistics over a subsequence for each user, then analyze the distribution of these statistics across the population. For example, we estimate the mean values $M'_1, \ldots, M'_n$ for each user's subsequence and characterize the distribution D that these values follow. We aim to demonstrate that more accurate estimations of $\{M'_1, \ldots, M'_n\}$ can lead to a more precise characterization of D. Formally, our theoretical framework establishes that accurate individual-level estimations inherently result in accurate crowd-level statistical inferences, as formalized in Theorem 5.

Theorem 5. *Given any user's true feature value A and estimated feature value $\hat{A}$, with the estimation error bounded by $|\hat{A} - A| \leq \beta$ (where β is a fixed constant), when the sample size N is sufficiently large, the maximum difference between the empirical distribution function $F_N(x)$ based on the estimated values $\hat{A}_i$ and the true distribution function $F(x)$ does not exceed η with probability at least $1 - \delta$, i.e.,*

$$P\left(\sup_x |F_N(x) - F(x)| \leq \eta\right) \geq 1 - \delta,$$

where $\eta > \beta$ is the given error bound and δ is the confidence parameter.

Proof. We need to analyze the difference between the empirical distribution function $F_N(x) = \frac{1}{N}\sum_{i=1}^{N} I(\hat{A}_i \leq x)$ constructed from the estimated values $\hat{A}_i$ and the true distribution function $F(x) = P(A \leq x)$.

To facilitate the analysis, we introduce the empirical distribution function based on the true values A_i, defined as $F_N^{\text{true}}(x) = \frac{1}{N}\sum_{i=1}^{N} I(A_i \leq x)$, and apply the triangle inequality to decompose the total error into two parts:

$$\sup_x |F_N(x) - F(x)| \leq \sup_x |F_N(x) - F_N^{\text{true}}(x)| + \sup_x |F_N^{\text{true}}(x) - F(x)|.$$

We first analyze the estimation error $\sup_x |F_N(x) - F_N^{\text{true}}(x)|$. Since $|\hat{A}_i - A_i| \leq \beta$, for any value of x, we have:

$$A_i \leq x \Rightarrow \hat{A}_i \leq x + \beta$$

and

$$\hat{A}_i \leq x \Rightarrow A_i \leq x + \beta$$

This implies that, for each point x:

$$F_N^{\text{true}}(x - \beta) \leq F_N(x) \leq F_N^{\text{true}}(x + \beta)$$

Therefore,

$$\begin{aligned} F_N(x) - F_N^{\text{true}}(x) &\leq F_N^{\text{true}}(x + \beta) - F_N^{\text{true}}(x) \\ F_N^{\text{true}}(x) - F_N(x) &\leq F_N^{\text{true}}(x) - F_N^{\text{true}}(x - \beta) \end{aligned}$$

Since the growth rate of an empirical distribution function does not exceed 1, we have:

$$\sup_x |F_N(x) - F_N^{\text{true}}(x)| \leq \beta$$

Next, we examine the convergence error $\sup_x |F_N^{\text{true}}(x) - F(x)|$. According to the DvoretzkyKieferWolfowitz (DKW) inequality, for any $\epsilon > 0$:

$$P\left(\sup_x |F_N^{\text{true}}(x) - F(x)| > \epsilon\right) \leq 2e^{-2N\epsilon^2}$$

Taking $\epsilon = \eta - \beta$ and setting $2e^{-2N(\eta-\beta)^2} \leq \delta$, we obtain:

$$N \geq \frac{\ln(2/\delta)}{2(\eta - \beta)^2}$$

When the sample size satisfies the above condition, we have:

$$P\left(\sup_x |F_N^{\text{true}}(x) - F(x)| \leq \eta - \beta\right) \geq 1 - \delta$$

Combining these results, we obtain:

$$P\left(\sup_x |F_N(x) - F(x)| \leq \beta + (\eta - \beta)\right) \geq 1 - \delta$$

$$P\left(\sup_x |F_N(x) - F(x)| \leq \eta\right) \geq 1 - \delta$$

This proves that when the sample size N is sufficiently large and satisfies $N \geq \frac{\ln(2/\delta)}{2(\eta-\beta)^2}$, the maximum difference between the empirical distribution function $F_N(x)$ constructed from the estimated values and the true distribution function $F(x)$ does not exceed η with probability at least $1 - \delta$. □

Extension to other mechanisms. Our methods extend beyond the SW mechanism to other numerical LDP mechanisms, including Laplace [9], SR [7], and PM [32]. Applying our approach to these mechanisms requires two key modifications. First, we normalize the original data to $[-1, 1]$ to accommodate the input requirements of these mechanisms. Under this setting, the Laplace mechanism adds noise from $Lap(2/\epsilon)$. Accordingly, the clipping process in IPP and APP must also be adjusted to $[-1, 1]$. Second, in CAPP, different mechanisms require specific clip intervals $[l, u]$. Due to space limitations, we omit the detailed discussion of these settings.

However, these mechanisms show inferior performance compared to SW. This is primarily because SW provides bounded perturbation results within $(-1/2, 3/2)$, regardless of the privacy budget. In contrast, PM mechanism with privacy budget $\epsilon = 0.01$ leads to perturbation in $[-400, 400]$. Laplace mechanism generates perturbations well beyond $[-1, 1]$ even with small noise. SR mechanism, limited to outputs $\{-1, 1\}$, loses substantial temporal information. These limitations explain our focus on the SW mechanism as the primary perturbation mechanism in this paper.

Extension to high-dimensional time series data. Our scheme can be extended to high-dimensional time series data, such as location information and trajectory data publication. We consider each dimensional time series independently and apply our methods separately. To satisfy the composition theorem of differential privacy, we employ privacy budget splitting and sampling techniques. Specifically, for d-dimensional time series data collection, we propose two approaches:

- Budget-Split (BS): At any given time slot, users upload data for all dimensions, with each data point allocated a privacy budget of $\epsilon/(dw)$. While this approach involves uploading more data points, it results in higher noise perturbation for each point.
- Sample-Split (SS): At any given time, users upload information for only one dimension, with each data point allocated a privacy budget of ϵ/w. In this approach, each window can only upload w/d data points, but the noise perturbation for each point is lower.

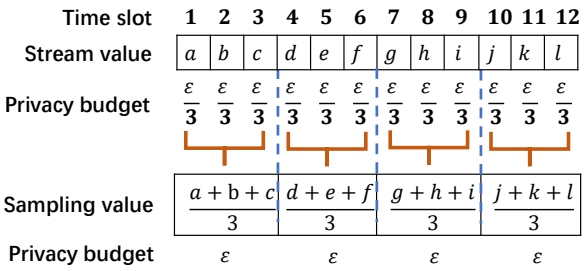


Fig. 3: Illustration of the sampling

V. Perturbation Parameterization Algorithms with Sampling

In this section, we combine sampling and perturbation parameterization (PP) algorithms to help enhance the accuracy of mean statistics while ensuring stream data publication remains satisfactory. Sampling approaches usually select a few values for reporting to increase the per-value privacy budget. However, if the number of values uploaded is too small, it may degrade the utility of stream data publication. To address this, we provide a perturbation parameterization sampling method that determines the optimal number of samples.

Sampling Design. Let n_s represent the number of samples for query $X_{(i,j)}$. To ensure w-event privacy in the sliding window model: We divide the query interval $[i, j]$ into n_s segments, each containing $\lfloor \frac{j-i+1}{n_s} \rfloor$ time slots[1]. For each segment, we sample one value at a predetermined position. The sampling positions remain consistent across different windows to maintain privacy guarantees.

A naive approach of simply uploading the value at the predetermined position to the data collector would result in information loss. To maximize the uploaded information while maintaining w-event local differential privacy, we upload the mean value of each segment. The user's original sampled values are then:

$$S = \{s_1, \ldots, s_r, \ldots, s_{n_s}\}.$$

where s_r is the mean value for the r-th segment.

Let us illustrate the sampling process with an example in Figure 3, where $w = 3$. Without sampling, each value must be uploaded individually with a privacy budget of $\epsilon/3$. In our sampling approach, we upload one aggregated value for every three consecutive values by computing their mean. This strategy allows us to increase the privacy budget for each uploaded value from $\epsilon/3$ to ϵ. Users then generate and perturb these sampled mean values using our parameterized perturbation algorithms before transmission to the data collector.

[1]When $(j - i + 1)$ is not divisible by n_s, the remaining time slots are assigned to the last segment.

After going through our perturbation parameterization algorithms, the perturbed values are:

$$S' = \{s'_1, \ldots, s'_r, \ldots, s'_{n_s}\}.$$

To restore the perturbed value for all time slots, we have:

$$X' = \{\overbrace{s'_1, \ldots, s'_1}^{\lfloor \frac{j-i+1}{n_s} \rfloor \text{ times}}, \ldots, \overbrace{s'_{n_s}, \ldots, s'_{n_s}}^{\lfloor \frac{j-i+1}{n_s} \rfloor \text{ times}}\}.$$

Algorithm 3 Perturbation Parameterization Sampling

Input: Privacy budget ϵ, window size w, time interval $[i, j]$
Output: $\hat{X}_{(i,j)} = \{\hat{x}_i, \ldots, \hat{x}_t, \ldots, \hat{x}_j\}$

1: Divide the $[i, j]$ into n_s segments
2: γ=min$\{\lfloor \frac{j-i+1}{n_s} \rfloor, w\}$, $\epsilon_w = \epsilon/\gamma$, $X' = \emptyset$
3: **for** each segment r **do**
4: Set x_r^I as the mean of current segment
5: $x'_r = PP(x_r^I)$ ▷ Same as PP, APP, CAPP
6: $X' = X' \sqcup \{x'_r\}^{\lfloor \frac{j-i+1}{n_s} \rfloor}$
7: **end for**
8: **return** $X'_{(i,j)} = \{x'_i, \ldots x'_t, \ldots, x'_j\}$

The procedure of PP-S. By integrating sampling techniques into Perturbation Parameterization methods (IPP, APP, CAPP), we propose the Perturbation Parameterization Sampling (PP-S) algorithm. The Algorithm 3 begins by dividing the time interval $[i, j]$ into n_s segments (line 1). It then determines the privacy budget per segment ϵ_w and initializes an empty set X' (line 2). For each segment r, the algorithm first calculates the mean value x_r^I of the current segment (line 4), and applies the Perturbation Parameterization (PP) method to perturb x_r^I (line 5). The perturbed value is then replicated to match the segment size and added to set X' (line 6). Finally, the algorithm returns the estimated stream data $X'_{(i,j)}$ (line 8).

The choice of n_s. As the number of samples n_s decreases, each uploaded value gets a larger privacy budget, allowing more accurate mean estimation. However, a smaller n_s compromises our ability to capture the stream characteristics effectively. Our goal is to determine an optimal n_s that balances accurate mean estimation with effective stream data publication.

Specifically, we will then utilize the distribution of the variance [24] of these n_s values to help determine the final size of n_s. Given w with a total privacy budget of ϵ, let $Var(n_s, \epsilon)$ denote the variance for the sample variance after sampling n_s times. The reason for using the **variance of the sample variance**, rather than the **sample variance** itself, is that it better reflects the variability and instability of the sample variance statistic under repeated random sampling. By multiplying the additional factor n_s, we amplify the impact of sampling size, and thus our objective function is:

$$\text{argmin}_{n_s} \, n_s Var(n_s, \epsilon). \tag{12}$$

Since $n_s \in \{1, \ldots, j - i + 1\}$ is an integer, we can only enumerate all possible values of n_s and find the one that minimizes the objective function.

Next, we detail the calculation of $Var(n_s, \epsilon)$. According to [24], $SW(x)$ is independent and identically distributed, then we have:

$$Var(n_s, \epsilon) = \frac{1}{n_s}\left(\mu_4 - \frac{\sigma^2(n_s - 3)}{n_s - 1}\right), \tag{13}$$

where σ^2 is the corresponding variance, μ_4 is the corresponding fourth central moment, which are all parameters related to ϵ. Due to unknown stream data, we examine $SW(x)$ at $x = 1$ (maximum variance case) and we obtain the expectation μ as follows:

$$\begin{aligned}\mu &= \int_{-b}^{1+b} SW(x)\, dx \\ &= \frac{q(2b - 4bx + 1)}{2} + 2bpx = 2b(p - q)x + qb + \frac{q}{2} \\ &= 2bp - bq + \frac{q}{2}.\end{aligned}$$

Next, we derive the variance σ^2, which is computed by the following equation:

$$\begin{aligned}\sigma^2 &= \int_{-b}^{1+b} (x - \mu)^2 SW(x)\, dx \\ &= \frac{2b^3 p}{3} - \frac{b^3 q}{3} - 4b^2p^2 + 4b^2pq - b^2q^2 + b^2q \\ &\quad - 2bpq + 2bp + bq^2 - \frac{2bq}{3} - \frac{q^2}{4} + \frac{q}{3}.\end{aligned}$$

Finally, we determine the fourth central moment μ_4, given by the equation:

$$\begin{aligned}\mu_4 &= \int_{-b}^{1+b} (x - \mu)^4 SW(x)\, dx \\ &= \frac{q}{5} + 2bp - bq - q\mu + 4b^3p + \frac{2b^5 p}{5} + 2b^2q \\ &\quad - 2b^3q + b^4q + 2q\mu^2 - 2q\mu^3 + q\mu^4 + 12bp\mu^2 - 8bp\mu^3 \\ &\quad - 8b^3p\mu + 2bp\mu^4 - 6bq\mu^2 - 6b^2q\mu + 4bq\mu^3 + 4b^3q\mu \\ &\quad + 4b^3p\mu^2 + 6b^2q\mu^2 - 8bp\mu + 4bq\mu.\end{aligned}$$

Guidelines for selecting the value of n_s. We provide heuristic guidelines for choosing n_s based on the distributional properties of the data. When n_s increases monotonically in Equation 12, $Var(n_s, \epsilon)$ must eventually exhibit a decreasing pattern to achieve its minimum. According to the definition of sample variance in Equation 13, for relatively small values of n_s, the objective function is primarily determined by the fourth moment μ_4, as the convergence rate of the sample fourth moment substantially exceeds that of $1/n$.

For heavy-tailed distributions (e.g., Cauchy distribution), $Var(n_s, \epsilon)$ tends to grow without bound as n_s increases. In such scenarios, selecting a relatively small n_s is recommended to prevent the potential explosion of $Var(n_s, \epsilon)$.

For light-tailed distributions (e.g., normal and uniform distributions), extreme values have limited influence on the fourth moment. Consequently, the fourth moment typically exists and $Var(n_s, \epsilon)$ demonstrates rapid convergence with increasing n_s. Although the behavior of $Var(n_s, \epsilon)$ may be complex for small n_s, an optimal n_s that minimizes Equation 13 typically occurs near the point where $Var(n_s, \epsilon)$ begins to decrease and stabilize. In these cases, selecting a moderate

value of n_s represents a robust choice.

Theorem 6 proves that the Perturbation Parameterization algorithm with sampling satisfies w-event LDP.

Theorem 6. *Let X and Y be any two w-neighboring subsequences, and let S' be an arbitrary output from the Perturbation Parameterization algorithm with sampling (PP-S), where PP-S represents IPP, APP, or CAPP. For a total of n_w sampled values in each window, with each sampled value allocated a privacy budget of $\frac{\epsilon}{n_w}$, the PP-S mechanism achieves w-event local differential privacy.*

Proof. Define $F = (f_1, \ldots, f_{n_w})$ as the sampling result from the stream $X = (x_1, x_2, \ldots, x_w)$, and $G = (g_1, \ldots, g_{n_w})$ as the sampling result from the stream $Y = (y_1, y_2, \ldots, y_w)$. Let $S' = (s'_1, s'_2, \ldots, s'_{n_w})$ denote the perturbed output sequence and S^r its reverse sequence. Applying these definitions, Equation 9 can be rewritten as:

$$\begin{aligned}
&\frac{P\{PP\text{-}S(f_1, f_2, ..., f_{n_w}) = S'\}}{P\{PP\text{-}S(g_1, g_2, ..., g_{n_w}) = S'\}} \\
=&\frac{P\{PP\text{-}S(f_{n_w}, f_{n_w-1}, ..., f_1) = S^r\}}{P\{PP\text{-}S(g_{n_w}, g_{n_w-1}, ..., g_1) = S^r\}} \\
=&\frac{P(s'_{n_w}|f^I_{n_w}) \ldots P(s'_2|f^I_2) P(s'_1|f_1)}{P(s'_{n_w}|g^I_{n_w}) \ldots P(s'_2|g^I_2) P(s'_1|g_1)} \\
\leq&(\frac{p}{q})^{n_w} \leq (e^{\epsilon/n_w})^{n_w} \leq e^{\epsilon}.
\end{aligned}$$

□

VI. Experimental Results

In this section, we conduct experiment to validate the effectiveness of our proposed solutions.

A. Experimental Setup

The experiments are conducted on a PC equipped with an AMD Ryzen 7 2700X eight-core processor, 64GB RAM, and Windows 10, using MATLAB R2019b. The experiments are conducted 100 times and the results were subsequently averaged. All datasets and code are available online[2].

1) Comparison algorithms: We categorize the compared algorithms into two sets: those without sampling and those with sampling. For the first set, we analyze several approaches, including a naive algorithm (**SW-direct**) that directly applies the SW perturbation method to each value, and ToPL [33], a state-of-the-art method for mean estimation from subsequences. We also compare BA-SW [17], [25], which integrates budget absorption with the SW mechanism to preserve the privacy budget by skipping the transmission of minimally changed data points. Our proposed algorithms in this set include IPP, APP, and CAPP. For the second set, involving sampling, we compare a naive approach (**Sampling**) that samples stream values before applying the SW mechanism, alongside our proposed methods APP-S and CAPP-S.

[2]https://github.com/RONGDUGithub/CAPP

As for the smoothing step, it is necessary to determine a smoothing window size. A larger window size offers advantages in estimating the mean; however, it may result in unfavorable outcomes for stream data publication. In our experimental setting, we choose the smoothing window size of 3.

2) Utility metrics: We classify our performance evaluations into three categories: (a) mean estimation with Mean Squared Error (MSE) assessment [18], (b) direct stream data release evaluated by cosine distance [28], and (c) distribution analysis of subsequence means using Wasserstein Distance [27].

MSE. The Mean Squared Error, denoted as MSE, is a commonly used metric to measure the average squared difference between predicted values $\hat{y}_i$ and true values y_i in regression tasks. Given a set of n pairs of predicted and true values, the MSE is calculated as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2.$$

Cosine distance. The cosine distance [28], denoted as $d_{\cos}$, measures the dissimilarity between two vectors in a vector space. Given two vectors $\mathbf{u}$ and $\mathbf{v}$, the cosine distance is calculated as:

$$d_{\cos}(\mathbf{u}, \mathbf{v}) = 1 - \frac{\mathbf{u} \cdot \mathbf{v}}{|\mathbf{u}||\mathbf{v}|}.$$

where $d_{cos}\cdot$ represents the dot product of the vectors, and $|\cdot|$ denotes the Euclidean norm. If the cosine distance is high (e.g., close to 1), indicating that the vectors are dissimilar.

Wasserstein Distance. Also known as Earth Mover's Distance (EMD), it measures the minimum cost of transforming one probability distribution into another. The Wasserstein distance is computed as the sum of absolute differences between two empirical CDFs F and G, as shown below:

$$W(F, G) = \sum_{i=1}^{n} |F_i - G_i|.$$

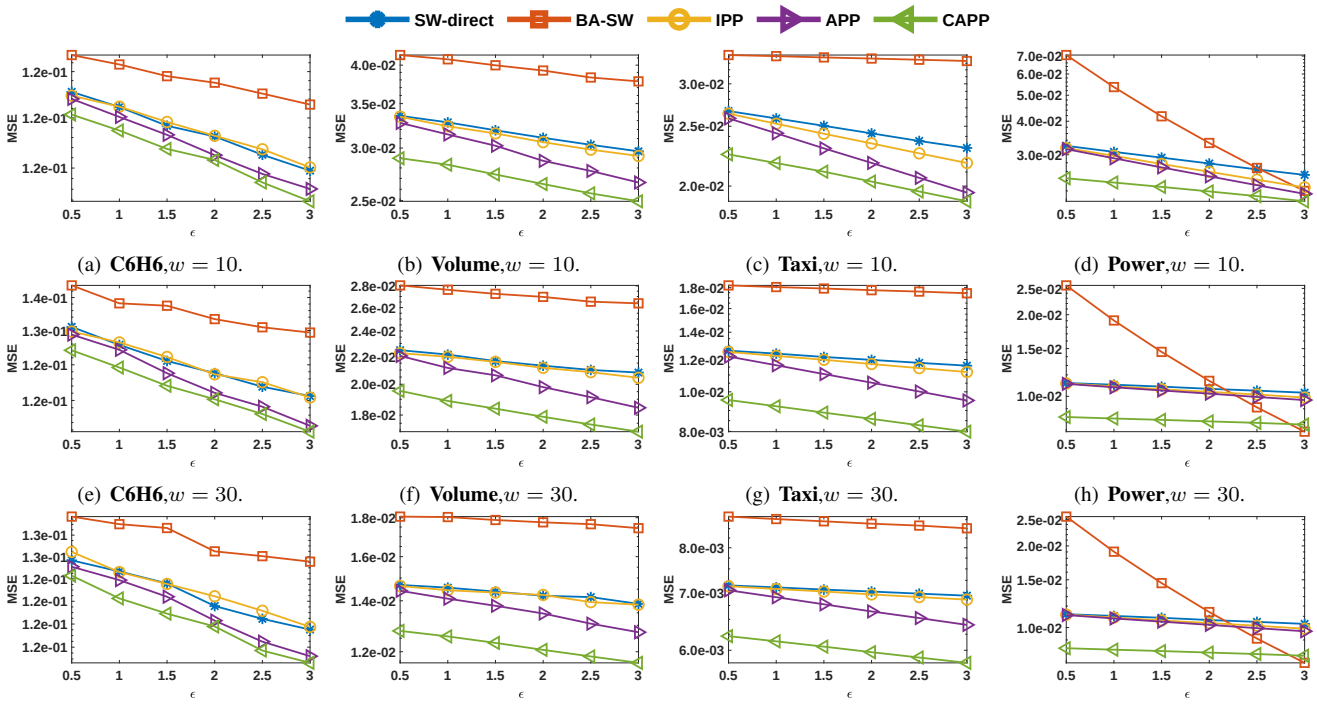
A smaller Wasserstein distance indicates higher similarity between the distributions.

3) Datasets: The experiments are conducted on four real-world datasets. The **Volume** and **C6H6** datasets each consist of a single data stream from a single user, while the **Taxi** and **Power** datasets contain multiple data streams from multiple users. For the latter two datasets, we perform both user-level and crowd-level statistical analysis.

Volume[3]**.** The dataset comprises hourly measurements of westbound traffic volume from the Minnesota Department of Transportation (MNDoT) Automatic Traffic Recorder (ATR) station 301, strategically situated midway between Minneapolis and St. Paul along Interstate 94. The data stream contains a total of 48204 valid entries.

C6H6[4]**.** The dataset encompasses 9,358 instances of hourly averaged data collected by a suite of five metal oxide chemical sensors integrated into an Air Quality Chemical Multisensor Device, spanning from March 2004 to February

[3]https://archive.ics.uci.edu/dataset/492/metro+interstate+traffic+volume
[4]https://archive.ics.uci.edu/dataset/360/air+quality



(a) **C6H6**,$w = 10$. (b) **Volume**,$w = 10$. (c) **Taxi**,$w = 10$. (d) **Power**,$w = 10$.

(e) **C6H6**,$w = 30$. (f) **Volume**,$w = 30$. (g) **Taxi**,$w = 30$. (h) **Power**,$w = 30$.

(i) **C6H6**,$w = 50$. (j) **Volume**,$w = 50$. (k) **Taxi**,$w = 50$. (l) **Power**,$w = 50$.

Fig. 4: MSE comparison w.r.t. ϵ for perturbation parameterization based algorithms vs SW-direct

2005. We focus on the subset of data related to the fluctuations in benzene concentration levels.

Taxi[5]**.** The dataset captures the real-time trajectories of 10,357 taxis in Beijing, recorded from February 2 to February 8, 2008. We focused on extracting the latitude information for each taxi at 1307 specific timestamps from 1500 drivers.

Power[6]**.** This dataset is sourced from the UCR Time Series Data Mining Archive. It contains the power usage data of 25,562 electrical devices, with each time series consisting of 96 stream values.

B. Overall Results for Perturbation Parameterization Algorithms

1) The results for mean estimation:

Comparison of SW-based algorithms and ToPL. In Table I, our comparative analysis primarily focuses on two categories of algorithms. The first category encompasses those based on the SW mechanism, which includes the SW-direct, IPP, and APP algorithms. The second category is the ToPL algorithm [33], which first uses the SW algorithm to remove outlier data and obtain a reasonable data range, then perturbs the data using the HM mechanism [32].

In Table I, we find that the MSE of ToPL is more than 100 times larger than others for mean estimation. This is because the perturbation threshold mechanism of SW is much smaller than that of HM. For example, when the privacy budget per window is 1, and there are 20 time slots, the privacy budget allocated to each time slot is 0.05. The perturbation threshold for SW is mapped from [0,1] to [-0.4836, 1.4836]. In contrast, the HM perturbation threshold is mapped from [-1,1] to [-80,80]. Thus, it is evident that ToPL incurs a larger error with a smaller privacy budget. Furthermore, the order of magnitude of this error increases exponentially as epsilon decreases. Due to the significant discrepancy of ToPL's results compared to PP methods, we will not include them in our subsequent experimental result figures.

TABLE I: Results for ToPl with SW-based algorithms

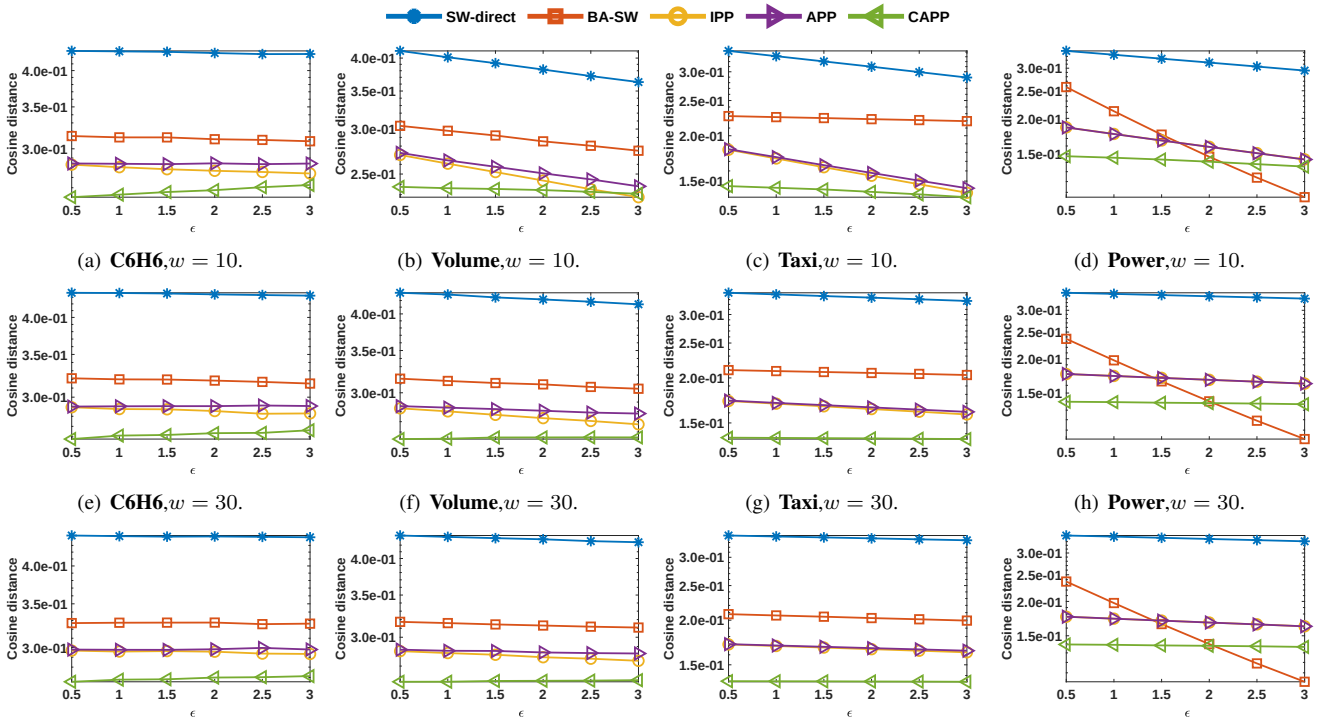
MSE	w	SW-direct	IPP	APP	ToPL
C6H6 $\epsilon = 1$	20	0.131	0.131	0.129	25.214
	40	0.125	0.126	0.125	51.613
	60	0.124	0.124	0.123	80.070
Taxi $\epsilon = 1$	20	1.28E-04	1.25E-04	1.19E-04	0.043
	40	4.9E-05	4.8E-05	4.6E-05	0.132
	60	3.7E-05	3.6E-05	3.5E-05	0.221

Comparison on PP algorithms and SW-direct & BA-SW. In Figure 4, we compare MSE for mean estimation among the SW-direct, BA-SW and our PP algorithms (IPP, APP, and CAPP). We compute the average over 50 randomly sampled time subsequences with length w.

We observe that in most subfigures, the MSE of BA-SW is the largest compared to our perturbation parameterization algorithms, with SW-direct performing as the second worst. The utility of APP is better than that of IPP, as APP leverages more perturbation information from the stream values, resulting in smaller errors in mean estimation, as demonstrated by Lemma IV.2. Notably, CAPP achieves the

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

[6]https://www.cs.ucr.edu/%20eamonn/time%20series%20data/



(a) **C6H6**,$w = 10$. (b) **Volume**,$w = 10$. (c) **Taxi**,$w = 10$. (d) **Power**,$w = 10$.

(e) **C6H6**,$w = 30$. (f) **Volume**,$w = 30$. (g) **Taxi**,$w = 30$. (h) **Power**,$w = 30$.

(i) **C6H6**,$w = 50$. (j) **Volume**,$w = 50$. (k) **Taxi**,$w = 50$. (l) **Power**,$w = 50$.

Fig. 5: Cosine distance comparison w.r.t. ϵ for perturbation parameterization algorithms vs SW-direct

best experimental results, which can be attributed to selecting a more suitable range for the input values. Under extremely limited privacy budgets, we reduce the mechanism's sensitivity by clipping the input values to a smaller range and normalizing them, thereby producing perturbed values with better distinguishability. This superior performance is consistent across different window sizes w, as shown in Figures 4 (b), (f), and (j).

It is worth noting that in Figures 4 (d), (h), and (l), the BA-SW algorithm performs best on the **Power** dataset when ϵ is relatively large. This is because many subsequences in the **Power** dataset are entirely composed of a unique constant value. BA-SW effectively identifies points with minimal variation and directly reuses previous values. By conserving the privacy budget, it introduces less noise, thereby improving the utility of subsequent uploads.

2) The results for stream data publication: In Figure 5, the basic setup is the same as that in Figure 4. We then compare the cosine distance for different algorithms, where a smaller cosine distance indicates that the estimated data stream is closer to the ground truth.

We can see that the SW-direct's cosine distance is the largest compared to our perturbation parameterization algorithms, indicating the worst performance. This is because our algorithms utilize the perturbation of previous data to reduce errors. We can see that CAPP demonstrates the best performance among the perturbation parameterization algorithms. This is because the method uses perturbation information and sets an appropriate $[l, u]$, further expanding the advantage. It is noteworthy that for **C6H6**, the utility of CAPP increases when ϵ increases. This is because the design of l, u considers the worst case scenario, which may lead to improper selection of $T_{(e_p,e_d)}$ in Equation 11 for different datasets. However, we can still see that CAPP achieves better utility than IPP and APP. In addition, we observe that IPP is slightly better than APP for stream data publication. This is because APP is influenced by multiple data points, which may result in poor estimates of the true characteristics of the data stream. In fact, APP is more suitable for mean estimation, as can be seen in Figure 4. In contrast, IPP preserves the information of individual data points better and thus achieves better utility than APP for stream data publication.

Similarly, our approaches maintain their superior performance across different datasets under the same parameter settings, as shown in Figures 5 (a)-(d), (e)-(h), and (i)-(l). Observing the varied w values specific to each dataset, as illustrated in Figures 5 (a), (e), and (i), it becomes clear that our perturbation parameterization algorithms sustain commendable performance across different w configurations. This demonstrates the robustness of our perturbation parameterization algorithms.

C. Overall Results for Sampling

1) The results for mean estimation: In this group of experiments, we compare different parameters across datasets, including the query length q, window size w, and privacy budget ϵ. To ensure the stability of mean estimation results, we randomly select 50 subsequences with length q and average them.

As shown in Figures 6, when q and w are fixed, the MSE decreases for all algorithms as the privacy budget increases.

Among them, sampling performs the worst because it only uploads information from sampled points. However, APP-S and CAPP-S outperform other non-sampling methods. This experiment demonstrates that our perturbation parameterization algorithms retain their advantage even with sampling, showcasing their superiority across different datasets and various w and q combinations.

2) The results for stream data publication: In Figure 7, we evaluate the performance of stream data publication by comparing the cosine distance between sampling and non-sampling algorithms. The results reveal different patterns from those observed in mean estimation (Figure 6). While sampling algorithms demonstrate significant advantages in mean estimation, they achieve similar performance levels in stream data publication. This pattern difference can be attributed to the reduced number of collecting values in each window due to sampling, which affects the two tasks differently: it helps reduce noise in mean estimation while still maintaining sufficient information for stream data publication. As illustrated in both figures, sampling-based algorithms perform well in mean estimation. While their performance in stream data publication is not as strong as CAPP, they still outperform APP.

D. Discussion on Generalizability

1) The results for crowd-level statistics: We extend our analysis from individual-level statistics to crowd-level statistics, as shown in Figure 8. The evaluation metric used is the Wasserstein distance between two distributions: the estimated distribution and the true distribution of means calculated from population subsequences. Figures 8 (a)-(d) present the results without sampling techniques. Among the methods, BA-SW performs the worst, followed by SW-direct, whereas CAPP outperforms APP and IPP. With sampling techniques (Figures 8 (e)-(h)), CAPP-S achieves the best performance, followed by APP-S, highlighting the synergy between our proposed methods and sampling. These results indicate that our methods not only provide accurate user-level time series statistics but also excel in estimating crowd-level statistics.

2) Generalizability to different mechanisms: We replace the SW mechanism with alternative numerical mechanisms (Laplace, SR, and PM), and the results are presented in Figure 9. The results demonstrate that our APP scheme consistently enhances performance across all mechanisms in both mean estimation and stream publication tasks. Notably, the SW mechanism demonstrates superior performance compared to all other mechanisms. This advantage arises from its bounded perturbation range of $(-\frac{1}{2}, \frac{1}{2})$, which preserves a substantial amount of information during the clipping process, irrespective of the ϵ value. In contrast, other mechanisms either sacrifice excessive information due to their broad perturbation domains or inherently discard significant data characteristics during their perturbation processes, as thoroughly analyzed in Section IV.C.

3) The results for high-dimensional time series: We extend our method to the analysis of high-dimensional time series. To evaluate its performance, we generate two datasets with $d = 5$ and $d = 12$ dimensions, where each dimension follows a sinusoidal function with varying frequency parameters. We process each dimension of the time series independently, applying privacy-preserving mechanisms separately to each one. To address the privacy budget constraints in this high-dimensional setting, we implement two strategies: Budget-Split (BS) and Sample-Split (SS). Figure 10 indicates that, while SW-SS and SW-BS perform the worst within their respective strategies, our APP and CAPP schemes significantly enhance the performance of both BS and SS approaches. However, the BS strategies (SW-BS, APP-BS, and CAPP-BS) outperform the SS strategies (SW-SS, APP-SS, and CAPP-SS), primarily because the SS approach suffers from reduced effectiveness caused by the limited number of data points per window introduced by sampling. This negative impact of sampling outweighs the disadvantages associated with splitting the privacy budget.

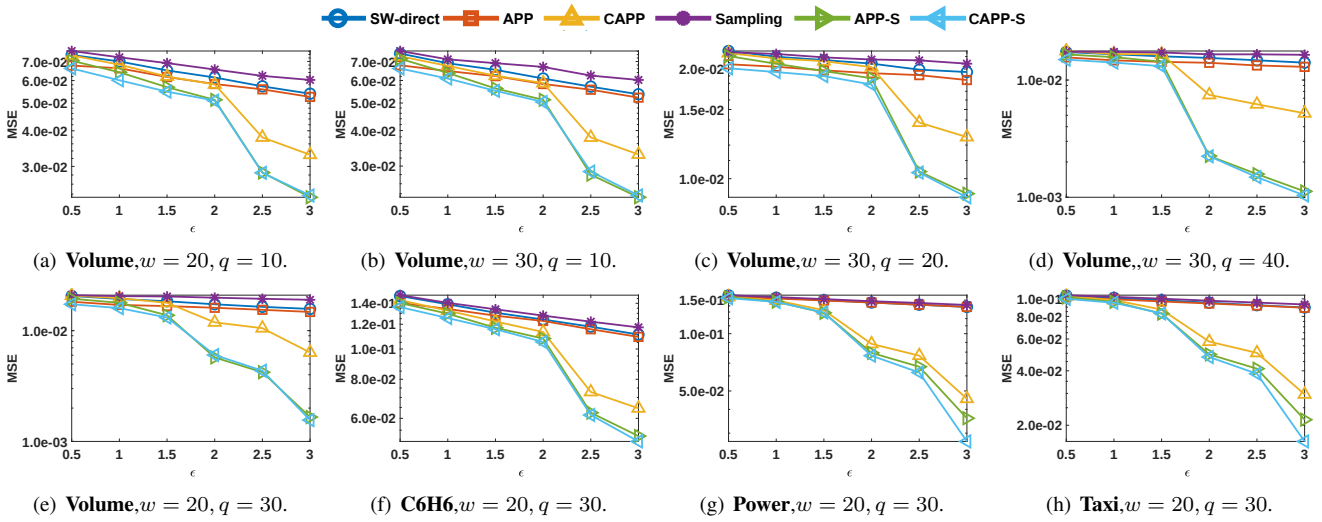
4) Sensitivity analysis on $[l, u]$: We represent l and u using δ (where $l = 0 - \delta$ and $u = 1 + \delta$), and evaluate performance for different δ values across four datasets: **Constant** (a time series with a constant value of $x = 0.1$), **Pulse** (zeros with a value of 1 inserted every five points), **Sinusoidal** (generated according to a sin(x) distribution), and **C6H6**. The experimental results, as shown in Figure 11, reveal consistent trends across all datasets: MSE decreases as ϵ increases. For a fixed ϵ, MSE follows a U-shaped curve as δ varies from -1 to 0.5, with the optimal δ (corresponding to the minimum MSE) differing across ϵ values. Generally, smaller ϵ values are associated with larger optimal δ values.

The MSE remains relatively stable in the vicinity of these optimal δ values. The recommended δ values (marked with stars), derived from Equation 11, are close to these optimal values and fall within regions of stable MSE. We recommend setting δ within the range $-0.25 \leq \delta \leq 0.25$, regardless of the original data distribution. For larger ϵ values, smaller δ values are preferable, whereas for smaller ϵ values, larger δ values are recommended.

VII. Related Work

Local Differential Privacy (LDP) [3], [6], [16] is an extension of Differential Privacy (DP) [8], [9], [22] tailored for individual users in distributed systems. LDP enables statistical analysis across data types, supporting mean, frequency, and distribution estimations. Within LDP, two primary privacy protection approaches exist: event-level [33] and user-level [1]. However, event-level LDP provides insufficient stream protection, while user-level LDP compromises utility through privacy budget partitioning. The w-event LDP [34] bridges this gap, offering a balanced framework for time series data protection.

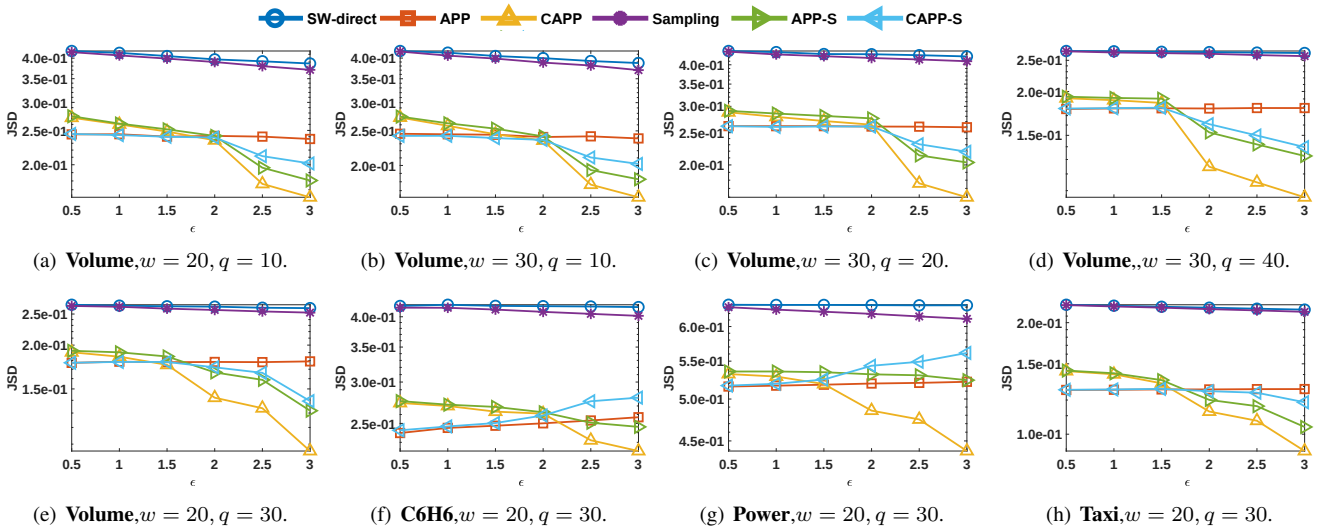
Several methods address streaming data under user-level LDP, primarily focusing on discrete data collection [10], [14], [35]. These approaches track statistical information by monitoring data changes to manage privacy budget consumption effectively. Specifically, [14] proposes a method that efficiently controls individual user contributions when the



(a) **Volume**,$w = 20, q = 10$. (b) **Volume**,$w = 30, q = 10$. (c) **Volume**,$w = 30, q = 20$. (d) **Volume**,,$w = 30, q = 40$.

(e) **Volume**,$w = 20, q = 30$. (f) **C6H6**,$w = 20, q = 30$. (g) **Power**,$w = 20, q = 30$. (h) **Taxi**,$w = 20, q = 30$.

Fig. 6: MSE comparison w.r.t. ϵ for sampling-based algorithms vs non-sampling algorithms



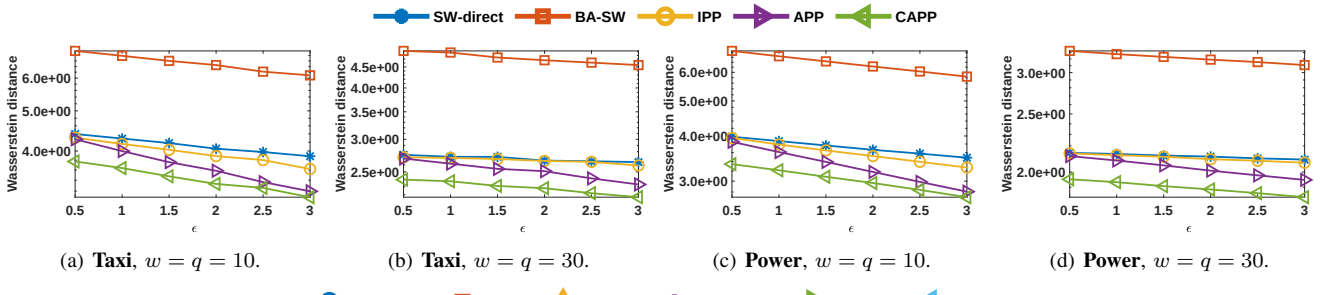
(a) **Volume**,$w = 20, q = 10$. (b) **Volume**,$w = 30, q = 10$. (c) **Volume**,$w = 30, q = 20$. (d) **Volume**,,$w = 30, q = 40$.

(e) **Volume**,$w = 20, q = 30$. (f) **C6H6**,$w = 20, q = 30$. (g) **Power**,$w = 20, q = 30$. (h) **Taxi**,$w = 20, q = 30$.

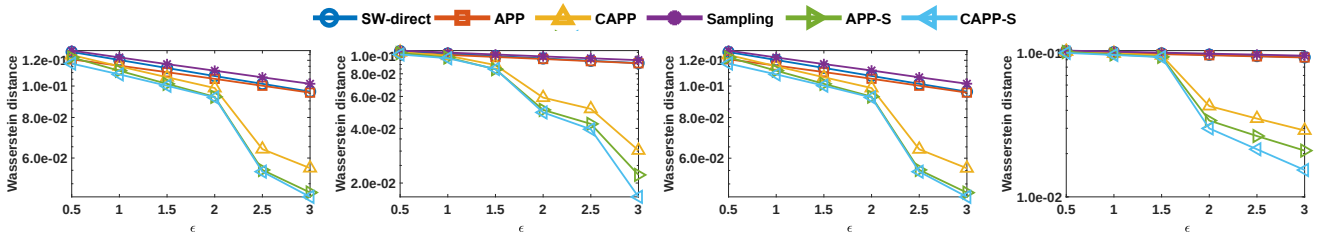
Fig. 7: Cosine Distance w.r.t. ϵ for sampling-based algorithms vs non-sampling algorithms

server's previous estimation is accurate, thereby conserving privacy budget. Erlingsson et al. [10] introduce a sanitization and reporting mechanism for continuous frequency estimation, where each user's value can change at most C times with increments/decrements of ± 1, and users randomly report one of their C changes. [35] presents DDRM, which employs binary trees to dynamically record temporal differences and suppresses privacy budget consumption during periods of data stability, taking advantage of the common occurrence of unchanged values in time series data. However, these methods are designed for discrete time series data and cannot be directly applied to continuous time series, which is the focus of our study.

In this paper, we address two types of tasks. The first type focuses on estimating individual-level statistics. Notable existing works include ToPL [33], Pattern LDP [34], and PrivShape [21]. ToPL [33] adopts event-level LDP constraints, which necessitate small privacy budgets for each timestamp, ultimately compromising utility in stream data publication. Pattern LDP [34] attempts to enhance utility by transmitting only significant changes with increased per-point privacy budgets, but this approach risks privacy leakage through the exposure of change points. PrivShape [21] only records key nodes, which may result in information loss when queried intervals contain few or no key nodes. Our work explores w-event level analysis as a balanced compromise between these extremes. The second type focuses on estimating statistical characteristics in crowd-level statistics. As demonstrated in LDP-IDS [25], this includes analyzing the distribution of population means. This method builds upon the BA [17] concept of utilizing general privacy budgets by not uploading points with minimal changes, thus conserving remaining privacy budgets. Additionally, it implements a sampling policy where users can upload their information in discontinuous windows, avoiding privacy budget splitting. Beyond simple time series, trajectory data represents a more complex form of time series, as explored in [17], [37], and [29]. These works maintain high utility

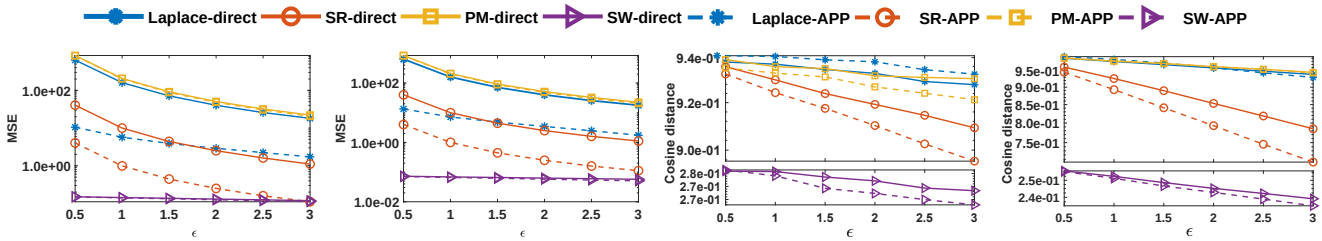


(a) **Taxi**, $w = q = 10$. (b) **Taxi**, $w = q = 30$. (c) **Power**, $w = q = 10$. (d) **Power**, $w = q = 30$.



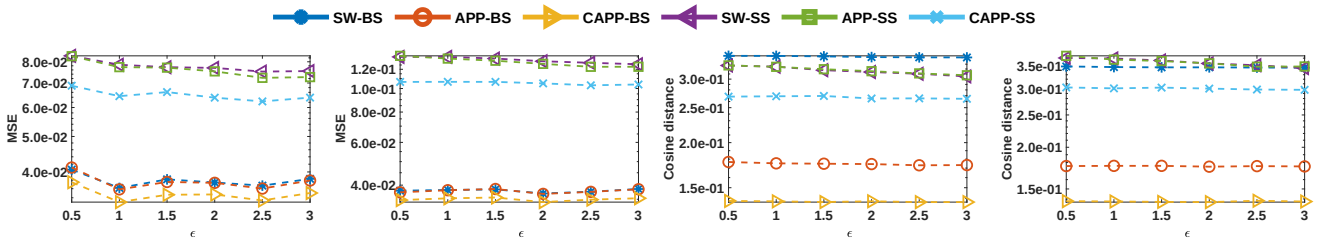
(e) **Taxi**, $w = 20, q = 10$. (f) **Taxi**, $w = 20, q = 30$. (g) **Taxi**, $w = 30, q = 10$. (h) **Taxi**, $w = 30, q = 40$.

Fig. 8: Wasserstein distance comparison of user mean distributions w.r.t. ϵ



(a) **C6H6**, MSE. (b) **Volume**, MSE. (c) **C6H6**, JSD. (d) **Volume**, JSD.

Fig. 9: Performance comparison of different LDP mechanisms with APP (Laplace, SR, PM, and SW)



(a) **Sin-data**, MSE, $d = 5$. (b) **Sin-data**, MSE, $d = 10$. (c) **Sin-data**, JSD, $d = 5$. (d) **Sin-data**, JSD, $d = 10$.

Fig. 10: Performance comparison of budget-split and sample-split in high-dimensional data

while satisfying local differential privacy requirements.

VIII. Conclusions

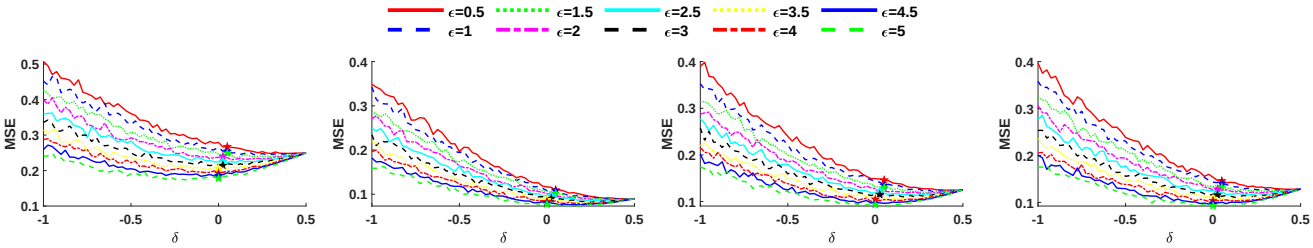
In this paper, we propose novel algorithms to collect and publish stream data under LDP. By effectively utilizing perturbations in two complementary ways, our methods significantly enhance utility compared to existing approaches. We present the IPP, APP, and CAPP algorithms and prove they satisfy w-event differential privacy. Moreover, we devise an optimized sampling scheme, further improving the accuracy of subsequence mean statistics. Experiments demonstrate the effectiveness of our techniques. This research opens a new direction for high-utility stream data analysis with privacy guarantees. Future work could extend this approach to handle more data types and apply it to emerging contexts like IoT and edge computing.

References

[1] Ergute Bao, Yin Yang, Xiaokui Xiao, and Bolin Ding. Cgm: an enhanced mechanism for streaming data collection with local differential privacy. *Proceedings of the VLDB Endowment*, 14(11):2258–2270, 2021.

[2] Eric Booth, Jeff Mount, and Joshua H Viers. Hydrologic variability of the cosumnes river floodplain. *San Francisco Estuary and Watershed Science*, 4(2), 2006.

[3] Rui Chen, Haoran Li, A Kai Qin, Shiva Prasad Kasiviswanathan, and Hongxia Jin. Private spatial data aggregation in the local setting. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 289–300. IEEE, 2016.



(a) **Constant**. (b) **Pulse**, MSE, $d = 10$. (c) **Sinusoidal**. (d) **C6H6**.

Fig. 11: Sensitivity analysis of δ on MSE across datasets, $w = q = 10$

[4] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. *arXiv preprint arXiv:1712.01524*, 2017.

[5] Dawn W Dowding, Marianne Turley, and Terhilda Garrido. The impact of an electronic health record on nurse sensitive patient outcomes: an interrupted time series analysis. *Journal of the American Medical Informatics Association*, 19(4):615–620, 2012.

[6] John C Duchi, Michael I Jordan, and Martin J Wainwright. Local privacy and statistical minimax rates. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 429–438. IEEE, 2013.

[7] John C Duchi, Michael I Jordan, and Martin J Wainwright. Minimax optimal procedures for locally private estimation. *Journal of the American Statistical Association*, 113(521):182–201, 2018.

[8] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[9] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[10] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2468–2479. SIAM, 2019.

[11] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[12] Ma Janice J Gumasing, Frances Jeann Charlize S Bermejo, Keisha Taranee C Elpedes, Lady Fatima E Gonzales, and Aaron Chastine V Villajin. Antecedents of waze mobile application usage as a solution for sustainable traffic management among gen z. 2023.

[13] AR Harris. Time series remote sensing of a climatically sensitive lake. *Remote Sensing of Environment*, 50(2):83–94, 1994.

[14] Matthew Joseph, Aaron Roth, Jonathan Ullman, and Bo Waggoner. Local differential privacy for evolving data. *Advances in Neural Information Processing Systems*, 31, 2018.

[15] James Joyce. Bayes theorem. 2003.

[16] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[17] Georgios Kellaris, Stavros Papadopoulos, Xiaokui Xiao, and Dimitris Papadias. Differentially private event sequences over infinite streams. *Proceedings of the VLDB Endowment*, 7(12):1155–1166, 2014.

[18] Erich L Lehmann and George Casella. *Theory of point estimation*. Springer Science & Business Media, 2006.

[19] Ninghui Li, Min Lyu, Dong Su, and Weining Yang. Differential privacy: From theory to practice. *Synthesis Lectures on Information Security, Privacy, & Trust*, 8(4):1–138, 2016.

[20] Zitao Li, Tianhao Wang, Milan Lopuhaä-Zwakenberg, Ninghui Li, and Boris Škoric. Estimating numerical distributions under local differential privacy. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 621–635, 2020.

[21] Yulian Mao, Qingqing Ye, Haibo Hu, Qi Wang, and Kai Huang. Privshape: Extracting shapes in time series under user-level local differential privacy. *arXiv preprint arXiv:2404.03873*, 2024.

[22] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103. IEEE, 2007.

[23] Thông T Nguyên, Xiaokui Xiao, Yin Yang, Siu Cheung Hui, Hyejin Shin, and Junbum Shin. Collecting and analyzing data from smart device users with local differential privacy. *arXiv preprint arXiv:1606.05053*, 2016.

[24] A Pranklin. *Introduction to the Theory of Statistics*. 1974.

[25] Xuebin Ren, Liang Shi, Weiren Yu, Shusen Yang, Cong Zhao, and Zongben Xu. Ldp-ids: Local differential privacy for infinite data streams. In *Proceedings of the 2022 international conference on management of data*, pages 1064–1077, 2022.

[26] Richard J Rossi. *Mathematical statistics: an introduction to likelihood based inference*. John Wiley & Sons, 2018.

[27] Ludger Rschendorf. The wasserstein distance and approximation theorems. *Probability Theory & Related Fields*, 70(1):117–129, 1985.

[28] Amit Singhal et al. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[29] Xinyue Sun, Qingqing Ye, Haibo Hu, Yuandong Wang, Kai Huang, Tianyu Wo, and Jie Xu. Synthesizing realistic trajectory data with differential privacy. *IEEE Transactions on Intelligent Transportation Systems*, 24(5):5502–5515, 2023.

[30] ADP Team et al. Learning with privacy at scale. *Apple Mach. Learn. J*, 1(8):1–25, 2017.

[31] Flemming Topse. On the glivenko-cantelli theorem. *Zeitschrift fr Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 14(3):239–250, 1970.

[32] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. Collecting and analyzing multidimensional data with local differential privacy. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 638–649. IEEE, 2019.

[33] Tianhao Wang, Joann Qiongna Chen, Zhikun Zhang, Dong Su, Yueqiang Cheng, Zhou Li, Ninghui Li, and Somesh Jha. Continuous release of data streams under both centralized and local differential privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1237–1253, 2021.

[34] Zhibo Wang, Wenxin Liu, Xiaoyi Pang, Ju Ren, Zhe Liu, and Yongle Chen. Towards pattern-aware privacy-preserving real-time data collection. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 109–118. IEEE, 2020.

[35] Qiao Xue, Qingqing Ye, Haibo Hu, Youwen Zhu, and Jian Wang. Ddrm: A continual frequency estimation mechanism with local differential privacy. *IEEE Transactions on Knowledge and Data Engineering*, 35(7):6784–6797, 2022.

[36] Mengxia Zhang and Lan Luo. Can consumer-posted photos serve as a leading indicator of restaurant survival? evidence from yelp. *Management Science*, 69(1):25–50, 2023.

[37] Yuemin Zhang, Qingqing Ye, Rui Chen, Haibo Hu, and Qilong Han. Trajectory data collection with local differential privacy. *arXiv preprint arXiv:2307.09339*, 2023.