MTSP-LDP: A Framework for Multi-Task Streaming Data Publication under Local Differential Privacy

CHANG LIU, MOE KLINNS Lab, Xi'an Jiaotong University, China
JUNZHOU ZHAO*, MOE KLINNS Lab, Xi'an Jiaotong University, China

The proliferation of streaming data analytics in data-driven applications raises critical privacy concerns, as directly collecting user data may compromise personal privacy. Although existing w-event local differential privacy (LDP) mechanisms provide formal guarantees without relying on trusted third parties, their practical deployment is hindered by two key limitations. First, these methods are designed primarily for publishing simple statistics at each timestamp, making them inherently unsuitable for complex queries. Second, they handle data at each timestamp independently, failing to capture temporal correlations and consequently degrading the overall utility. To address these issues, we propose MTSP-LDP, a novel framework for **M**ulti-**T**ask **S**treaming data **P**ublication under w-event LDP. MTSP-LDP adopts an *Optimal Privacy Budget Allocation* algorithm to dynamically allocate privacy budgets by analyzing temporal correlations within each window. It then constructs a *data-adaptive private binary tree structure* to support complex queries, which is further refined by cross-timestamp grouping and smoothing operations to enhance estimation accuracy. Furthermore, a unified *Budget-Free Multi-Task Processing* mechanism is introduced to support a variety of streaming queries without consuming additional privacy budget. Extensive experiments on real-world datasets demonstrate that MTSP-LDP consistently achieves high utility across various streaming tasks, significantly outperforming existing methods.

CCS Concepts: • **Security and privacy** → **Privacy-preserving protocols**; • **Information systems** → *Data management systems.*

Additional Key Words and Phrases: Data Streams, Local Differential Privacy

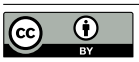
ACM Reference Format:
CHANG LIU and JUNZHOU ZHAO. 2026. MTSP-LDP: A Framework for Multi-Task Streaming Data Publication under Local Differential Privacy. *Proc. ACM Manag. Data* 4, 1, Article 55 (February 2026), 27 pages. https://doi.org/10.1145/3786669

1 Introduction

With the rapid development of the Internet of Things, massive real-time data streams have become ubiquitous in a variety of modern applications such as traffic management and intelligent parking [3, 4]. However, the collection and analysis of sensitive user data, such as real-time vehicle locations in a city and user purchasing history on e-commerce websites, pose significant privacy risks [1, 31, 36]. To protect user privacy, differential privacy (DP) [17] has emerged as a cornerstone, offering rigorous guarantees by injecting calibrated noise into user data. While centralized differential privacy (CDP) [6] relies on a trusted server to collect and perturb data, local differential privacy (LDP) [12, 16, 32–34, 38, 43, 45, 48, 51, 52] eliminates this requirement by applying perturbation

*Corresponding author

Authors' Contact Information: CHANG LIU, MOE KLINNS Lab, Xi'an Jiaotong University, Xi'an 710049, China, MollyLiu@stu.xjtu.edu.cn; JUNZHOU ZHAO, MOE KLINNS Lab, Xi'an Jiaotong University, Xi'an 710049, China, junzhou.zhao@xjtu.edu.cn.



ACM 2836-6573/2026/2-ART55
https://doi.org/10.1145/3786669

directly on the user side. This advantage has led to the deployment of LDP in large-scale systems, including Google's RAPPOR [22], Apple's iOS analytics [42], and Microsoft's telemetry services [13].

Researchers have proposed many privacy-preserving methods for streaming data, most of which focus on user-level DP for finite streams [23–25] or event-level DP for infinite streams [6, 18]. However, real-world applications usually run for long periods, producing infinite streams. Providing event-level privacy on such streams cannot satisfy the requirements for protecting arbitrary events. In contrast, providing user-level privacy requires adding infinite perturbations, which ultimately reduces the utility of the data. To address these issues, Kellaris et al. [28] proposed w-event DP to protect any sequence of events that occur in any w consecutive timestamps (i.e., a sliding window of size w), achieving a balance between privacy and utility for infinite streams.

However, existing w-event DP studies [39, 44, 49, 50] are primarily designed under the CDP setting. As an emerging solution for infinite data streams, w-event LDP [37] shows strong potential but remains underexplored. There are two key limitations that remain understudied and hinder its practical adoption. First, existing mechanisms fail to achieve efficient privacy budget utilization. While dynamic budget allocation is theoretically possible, empirical studies reveal that these strategies often perform worse than simple baselines, such as uniform allocation or random sampling [40]. Second, w-event LDP is severely limited in the types of analytical tasks it supports. While event-level and user-level DP methods have matured to support diverse analytical tasks including counting, range queries, and event monitoring [8, 9, 11], existing w-event methods remain limited to fixed-granularity statistical releases, particularly frequency histograms [30, 37]. This single-focus output makes it impossible to perform *multi-granularity* analysis on the same continuous stream. This rigidity is a critical barrier in real-world applications such as intelligent vehicle systems [41], where operators require insights at different *spatial* resolutions. For example, they require fine-grained insights, such as the specific vehicle count at a single intersection to manage traffic lights. Simultaneously, they must analyze coarse-grained data, such as the total traffic volume along an entire arterial road to identify bottlenecks.

To overcome these limitations and design a w-event LDP framework that supports concurrent multi-task processing, we identify four fundamental challenges to be addressed:

- **Spatial Imbalance in Data Distribution.** Real-world streaming data often exhibits significant spatial imbalance within individual timestamps. For instance, in intelligent transportation systems, traffic on arterial roads may far exceed that on side roads. Existing methods that uniformly partition the domain and apply the same noise scale to all regions can bury signals from sparse regions beneath the noise magnitude, obscuring underlying patterns and impairing downstream analytics.
- **Temporal Variation in Data Distribution.** The variation in stream distributions over time poses a significant challenge for allocating privacy budgets that can adaptively adjust to the degree of variation across the sliding window. Existing methods, often focusing solely on local information, fail to capture these dynamics, resulting in poor utility.
- **Privacy Budget Sharing across Multiple Tasks.** Streaming systems are often required to support multiple query tasks concurrently. As the number of concurrent tasks increases, the limited privacy budget must be divided among more tasks, causing a rapid decline in the budget allocated to each task and significantly degrading overall accuracy.
- **Low-Latency Constraints in Real-Time Processing.** Real-time streaming applications impose strict latency requirements. Even if a more accurate mechanism exists, it may be impractical if it cannot respond within tight deadlines. This necessitates a trade-off among privacy protection, estimation accuracy, and processing latency.

To address the above challenges, we propose MTSP-LDP, a new w-event LDP framework for infinite streams. Our main contributions are summarized as follows.

- To the best of our knowledge, this paper is the first to address the challenge of supporting multi-task, multi-granularity analytical tasks under w-event LDP. Specifically, we introduce the *Budget-Free Multi-Task Processing* mechanism, which enables concurrent support for different types of queries over infinite data streams. Notably, this mechanism operates without requiring modifications to other modules or the perturbation data reported by users. Crucially, it incurs no additional privacy cost.
- Our framework's high utility is achieved through two key technical innovations. First, the *Optimal Privacy Budget Allocation* mechanism adaptively allocates privacy budgets based on data variation across each sliding window of size w, ensuring that timestamps with greater variation receive more privacy budget. Second, the *Private Adaptive Tree Publication* mechanism improves accuracy by constructing a tree tailored to the underlying data distribution, departing from traditional fixed-partition histograms. It then applies cross-timestamp grouping and smoothing to further enhance estimation accuracy. Crucially, our mechanism resolves the prohibitive level-by-level interaction latency associated with traditional adaptive trees originally designed for static settings, thus enabling data-adaptive structures for streaming data.
- We conduct extensive experiments on four real-world datasets. Experimental results demonstrate that MTSP-LDP consistently outperforms state-of-the-art approaches designed for these specific tasks, achieving high utility under strict privacy guarantees.

2 Preliminaries and Problem Definition

2.1 Data Stream Model

We consider a distributed system consisting of a server and a set of n users, $U = \{u_1, \ldots, u_n\}$. Each user $u_i \in U$ reports a value $v_{it} \in \Omega$ at each discrete timestamp t, e.g., the user's location in a city. We assume users' values are from a finite domain of size d, denoted by $\Omega = \{\omega_1, \ldots, \omega_d\}$. All users' reported values at time t form a dataset D_t, which consists of n rows and d columns with $D_t[i][j] = 1$ if user u_i reported value ω_j at time t and $D_t[i][j] = 0$ otherwise. We allow a user u_i not to report her value at time t, i.e., u_i is inactive at t, and in this case, the i-th row of D_t is zero. Let n_t denote the number of active users at time t. Given D_t, the server can compute statistics of interest to publish. For example, the server may continuously publish a vector $\mathbf{c}_t = [c_{1t}, \ldots, c_{dt}]^\top$ at each time t, where c_{jt} denotes the number of users reporting the location $\omega_j \in \Omega$ at time t, and publishing $\mathbf{c}_t$ is useful for people to know the traffic congestion situation in a city at time t.

However, the user's value may be private (e.g., her location), and publishing statistics computed from this raw data can compromise privacy. For example, suppose that there is only one active user in the system. In this case, the published statistics directly reveal the user's private value. In addition, the server is not assumed to be trusted (e.g., it is compromised or contains software bugs), and it may leak users' private values. Therefore, privacy-preserving techniques are required to protect users' sensitive data from the server.

2.2 w-event LDP on Data Streams

To precisely define privacy on data streams, we review a useful concept called w-event LDP [37]. Let $V_{it} = (v_{i1}, \ldots, v_{it})$ denote user u_i's stream by time t.

Definition 1 (w-neighboring). *Let V_{it} and V'_{it} denote two streams of a single user u_i by time t. Let w denote a positive integer. V_{it} and V'_{it} are w-neighboring, if for each $v_{ir}, v_{is}, v'_{ir}, v'_{is}$ with $1 \le r \le s \le t, v_{ir} \ne v'_{ir}$ and $v_{is} \ne v'_{is}$, it holds that $s - r + 1 \le w$.*

In other words, we consider two versions of a user u_i's stream V_{it} and V'_{it}, and we say V_{it} and V'_{it} are w-neighboring if they have different values at timestamps fitting in a window of size up to w.

Definition 2 (w-event LDP). *Let $\mathcal{M}$ be a mechanism that takes as input a stream V_{it} of user u_i. Let O denote the set of all possible outputs of $\mathcal{M}$. We say that $\mathcal{M}$ satisfies w-event ϵ-local differential privacy (or, simply, w-event LDP) if for any w-neighboring streams V_{it}, V'_{it}, $\forall O \subseteq O$ and all t, it holds that*

$$\Pr[\mathcal{M}(V_{it}) \in O] \leq e^{\epsilon} \Pr[\mathcal{M}(V'_{it}) \in O].$$

The definition captures that a w-event LDP mechanism guarantees ϵ-LDP for each user within any window of size w. The parameter ϵ controls the privacy risk by injecting a proper amount of noise to user data [20]. A smaller ϵ indicates stronger privacy protection but also lowers the accuracy and utility of the published result as more noise is added to user data.

2.3 Frequency Oracle (FO) Under LDP

Frequency oracle (FO) protocols are common building blocks of many privacy-preserving techniques. An FO can estimate the frequency distribution of a private attribute while preserving privacy. Here we briefly introduce *optimized unary encoding* (OUE) [45], a widely used FO protocol that achieves ϵ-LDP with high estimation accuracy. OUE consists of the following steps.

- **Encoding.** Each user u_i encodes her private value $v_i \in \Omega$ into a one-hot binary vector $\mathbf{x}_i$ of length d, where the j-th bit $\mathbf{x}_i[j] = 1$ if and only if $v_i = \omega_j$, and 0 otherwise.
- **Perturbation.** Instead of directly submitting $\mathbf{x}_i$ to a server, OUE perturbs each bit of $\mathbf{x}_i$ independently. Specifically, if $\mathbf{x}_i[j] = 1$, it remains 1 with probability $p = 1/2$ and flips to 0 with probability $1 - p$. If $\mathbf{x}_i[j] = 0$, it flips to 1 with probability $q = \frac{1}{e^{\epsilon}+1}$ and remains 0 with probability $1 - q$. The perturbed vector $\mathbf{x}'_i$ is then sent to the server, thereby preserving users' privacy.
- **Aggregation.** The server aggregates the perturbed vectors from all users and estimates the frequency for each value ω_j. Let $\mathbf{y}[j]$ be the total number of perturbed reports with bit j set to 1. An unbiased estimate of the true frequency f_j is given by

$$\hat{f}_j = \frac{\mathbf{y}[j] - nq}{n(p - q)}, \quad j = 1, \ldots, d.$$

- **Estimation Error.** OUE satisfies ϵ-LDP, and the variance of the OUE estimator $\hat{f}_j$ is

$$var(\hat{f}_j) = \frac{4e^{\epsilon}}{n(e^{\epsilon} - 1)^2}, \quad j = 1, \ldots, d.$$

  This variance depends on the number of users n and the privacy budget ϵ. To simplify notation, in the following discussion, we will denote the variance of OUE by $var(\epsilon)$ while the number of users should be clear from context.

2.4 Private Binary Tree for Static Data

Private binary trees are widely used in many scenarios for different purposes under LDP [10, 26, 47]. For example, Chan et al. [10] first utilized a binary interval tree to represent a binary stream where each incoming binary value is assigned to a leaf node, enabling the efficient computation of time-range queries. Wang et al. [47] leveraged a hierarchy interval tree to support multi-dimensional queries in the static setting.

In this work, we use this data structure as summary statistics of user data D_t. As illustrated in Fig. 1, a private binary tree is a perfect binary tree defined on the value domain Ω. Each node of the tree is associated with an interval (or a set), that is the union of its child nodes' intervals. Nodes in the same level have disjoint intervals, and they form a partition of Ω. Each node is assigned with a *property*, e.g., the fraction of users holding a value in the node's interval (i.e., its frequency).

Formally, let $\mathbb{T}_t^d$ denote the tree built from user data D_t at time t, and let $\mathbb{T}_t^d[a]$ denote the property of node a. For example, $\mathbb{T}_t^d[a]$ in Fig. 1 denotes the fraction of users holding a value in set $\{0, 1\}$.

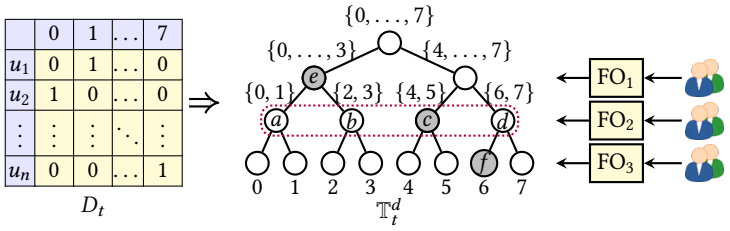


Fig. 1. Representing user data as a private binary tree, using FO to estimate each node's property in LDP, and performing a range query on interval $[0, 6]$.

While in the LDP setting, the server can not directly access user data D_t. To build a private binary tree of D_t, the server can leverage an FO. Specifically, we randomly partition users into $L - 1$ disjoint groups of equal size, where L is the number of levels of the tree. For each level except level 0 (which only has the root node representing domain Ω), the server invokes an FO whose domain size equals the number of nodes at that level, using privacy budget ϵ. Then, for each node in the level, the frequency estimate of FO is actually the fraction of users holding a private value in the node's interval. Because each user participates at exactly one level, by parallel composition the overall procedure still satisfies ϵ-LDP. For example, in Fig. 1, the server can invoke FO_2 in level 2 on a domain with 4 intervals, and the frequency estimates of FO_2 are estimates of $\mathbb{T}_t^d[x]$ for $x \in \{a, b, c, d\}$.

In the following discussion, we let $\hat{\mathbb{T}}_t^d$ denote the estimated private binary tree of D_t. Note that $\hat{\mathbb{T}}_t^d$ and $\mathbb{T}_t^d$ have the same structure except that node properties in $\hat{\mathbb{T}}_t^d$ are estimated using an FO. Because every FO is run for each level with the same number of users, the variance of $\hat{\mathbb{T}}_t^d[a]$ is the same, i.e., $var(\epsilon)$ when privacy budget is ϵ.

The estimated private binary tree $\hat{\mathbb{T}}_t^d$ can be used to answer range queries with high estimation accuracy. To answer a range query for a specified interval, we select a minimum number of nodes in the tree that cover the query interval. Node properties of this minimum cover can be used to answer the query. For example, in Fig. 1, if we want to estimate how many users holding a private value in $\{0, \ldots, 6\}$, we first find a minimum cover $\{e, c, f\}$ and then answer the query as $(\hat{\mathbb{T}}_t^d[e] + \hat{\mathbb{T}}_t^d[c] + \hat{\mathbb{T}}_t^d[f])n_t$.

While this structure effectively supports queries on static datasets, its direct extension to infinite streams entails significant limitations. A natural approach is to treat each time step independently and build a separate private tree for each D_t. However, this ignores the temporal correlations inherent in streaming data and requires composing the privacy loss over a long (potentially unbounded) sequence of releases, which leads to poor utility under our streaming privacy notion. Moreover, the tree partition is fixed and cannot adapt to time-varying data distributions; when the distribution changes over time, many nodes become either too sparse or too dense, resulting in large estimation errors. These limitations necessitate the development of our proposed framework, which builds on the above tree representation but incorporates dynamic budget allocation and data-adaptive mechanisms to handle evolving streams.

2.5 Queries on Data Streams

Given a data stream formed by user data at each timestamp, i.e., $\mathcal{D} = (D_1, D_2, \ldots)$, other than the histogram/frequency query extensively studied in previous work [28, 37](i.e., the query $\mathbf{c}_t/n_t$ defined in §2.1), we define several novel streaming query tasks.

Definition 3 (Counting Query). *Let $c_t(v)$ denote the number of users reporting the private value $v \in \Omega$ at time t, i.e., $c_t(v) \triangleq |\{u_i \in U : v_{it} = v\}|$. Given a value $v \in \Omega$ and a positive integer Δ, a counting query aims to report the total number of times users report the private value v in the most recent Δ timestamps. Formally,*

$$Q_t^c(v, \Delta) \triangleq \sum_{t'=t-\Delta+1}^{t} c_{t'}(v).$$

The counting query is a generalization of the histogram query and has widespread applications. Consider the following example.

Example 1. *Consider a vehicle dispatch management system in a city. At each time t, the system needs to publish the total number of times vehicles passed by a location v over the past ten minutes. Assuming a time granularity of one minute, then the query corresponds to $Q_t^c(v, 10)$.*

As a special case, when $\Delta = 1$, $Q_t^c(v, 1) = c_t(v)$ which is the number of users reporting v at time t, and $Q_t^c(v, 1)/n_t$ becomes the frequency of value v at time t.

Definition 4 (Range Query). *Given a value range $[v_1, v_2]$ (where $v_1 \le v_2$) and a positive integer Δ, a range query aims to report at each time t the total number of times users report private values in $[v_1, v_2]$ in the most recent Δ timestamps. Formally,*

$$Q_t^r([v_1, v_2], \Delta) \triangleq \sum_{t'=t-\Delta+1}^{t} \sum_{v \in [v_1, v_2]} c_{t'}(v)$$

The range query further generalizes the counting query by allowing the user value to be in a specified range. Consider the following example.

Example 2. *A disease monitoring center needs to publish daily the total number of infected people with age above 60 over the past week during the COVID pandemic. Assuming a time granularity of one day, then the query corresponds to $Q_t^r([60, \infty), 7)$.*

In many real-world scenarios, the server also wants to identify specific abnormal events in the stream through a class of queries known as event monitoring.

Definition 5 (Event Monitoring). *Let α and β be two functions, where α could be a counting query or a range query on the stream, and β is a Boolean function defined on the output of α. An event monitoring query, denoted by $Q_t^e(\alpha, \beta)$, aims to report at each time t a binary result where 1 denotes the monitored event occurs, and 0 otherwise.*

We illustrate the usefulness of this concept by the following example. Let $\Delta > 0$ denote a specified monitoring period, and we are interested in monitoring whether counts of a value v significantly increase in two consecutive monitoring periods. Then the two functions α and β can be formally defined as

$$\alpha(v, \Delta) = Q_t^c(v, \Delta) - Q_{t-w+1}^c(v, \Delta),$$
$$\beta(x, \vartheta) = \mathbf{1}_{x > \vartheta},$$

where $\mathbf{1}_B$ denotes an indicator function, and it returns 1 if the condition B is true, otherwise 0. The event monitoring query is a compound function on the stream, i.e., $Q_t^e(\alpha, \beta) = \beta \circ \alpha$, which reports

1 if there is a sudden increase of counts of value v in two consecutive monitoring periods. It is also straightforward to define α using a range query to monitor multiple user values. Therefore, we can use the defined event monitoring queries to monitor the sudden change in counts in Examples 1 and 2.

Remarks. The streaming queries proposed above are generalizations of existing works [11, 28] with the emphasis on data streams and aiming to achieve w-event LDP. Furthermore, for parameter Δ in these queries, we require $\Delta \leq w$ in order to satisfy w-event LDP.

2.6 Recent Methods to Achieve w-event LDP

State-of-the-art w-event LDP methods [37] focus on releasing statistical histograms at each timestamp. These methods can be categorized into the following four types based on their privacy budget allocation strategies.

- **LDP Budget Uniform (LBU)** applies a fixed ϵ/w budget to each timestamp in the window. However, as window size w increases, the allocated budget per timestamp becomes vanishingly small, introducing excessive noise and severely impairing utility.
- **LDP Sampling (LSP)** allocates the entire privacy budget ϵ to a single timestamp within the sliding window, providing high accuracy at that point while reusing its result to approximate the data for all other timestamps. This method performs well when the stream is stable but fails to adapt when the stream fluctuates. The approximation errors can become very large if subsequent stream data differ significantly from previous releases.
- **LDP Budget Distribution (LBD)** and **LDP Budget Absorption (LBA)** adopt dynamic budget allocation techniques, drawing inspiration from w-event CDP [28]. Both methods consist of two sub-mechanisms: private dissimilarity estimation and private strategy determination. The private dissimilarity estimation sub-mechanism computes the private dissimilarity between the current true statistics and the previous release based on perturbed user data collected via an FO with a fixed budget $\epsilon_{t,1} = \epsilon/(2w)$. The private strategy determination sub-mechanism decides whether to publish new statistics or reuse previous values by comparing the estimated dissimilarity with a potential publication error determined by the publication budget $\epsilon_{t,2}$. The allocation of $\epsilon_{t,2}$ differs between LBD and LBA. In LBD, the budget is distributed across timestamps requiring data publication in an exponentially decaying manner, and budget spent at timestamps outside the current window is reclaimed for reuse. In contrast, LBA first allocates the budget uniformly and then absorbs it at the timestamps that use approximation.
- Population Division Extensions. To improve the utility of LDP mechanisms, population division has emerged as a promising strategy [29, 37, 45]. Instead of splitting the privacy budget across timestamps, this paradigm partitions the user population, assigning each user to report at a specific timestamp using the entire privacy budget. This reduces estimation variance via the amplification-by-subsampling effect and enables more accurate data release within each reporting round. Based on this idea, LBU, LBD, and LBA can be extended to LPU (Population-based Uniform), LPD (Population-based Distribution), and LPA (Population-based Absorption), respectively. Notably, LSP intrinsically follows this paradigm. Population division is a general augmentation strategy that can be applied to any FO-based mechanism, including our proposed MTSP-LDP. While population division improves utility within each reporting round, it does not capture temporal correlations across timestamps, which are critical in streaming scenarios.

Remarks. Adaptive methods such as LBD and LBA, while designed to flexibly allocate privacy budgets based on data variations, have been shown to exhibit even lower utility in practice[40], failing to leverage the potential advantages of dynamic privacy budget allocation. Specifically, the private dissimilarity estimation sub-mechanism consumes half of the total privacy budget to collect perturbed data. This data is then discarded after estimating dissimilarity, halving the budget available for data publication. Additionally, these methods rely solely on single-timestamp information, neglecting the rich temporal correlations within the sliding window. Consequently, this strategy is insufficient, as the w-event LDP constraint requires the total budget ϵ to be allocated reasonably across all w timestamps, not just based on single-timestamp changes and the instantaneous budget cap.

3 The MTSP-LDP Framework

In this section, we propose a novel framework for **M**ulti-**T**ask **S**treaming data **P**ublication with w-event **LDP** guarantee (MTSP-LDP). MTSP-LDP is designed to efficiently answer streaming queries on infinite data streams and achieve w-event LDP.

3.1 Overview

At a high level, MTSP-LDP follows the *dissimilarity guided publication* framework, first proposed for the w-event CDP setting [28] and later extended to the w-event LDP setting [37]. The main idea of this framework is that, the server checks at every timestamp whether it is more beneficial to approximate the current stream statistics with the last released statistics, than to publish newly computed stream statistics with necessary noise. MTSP-LDP further enhances this framework by proposing an *optimal budget allocation strategy* to improve estimation accuracy and a *private binary tree structure* to enable multi-task streaming queries. In more detail, MTSP-LDP consists of four mechanisms, i.e., private dissimilarity estimation, optimal privacy budget allocation, private adaptive tree publication, and budget-free multi-task streaming query, as illustrated in Fig. 2.

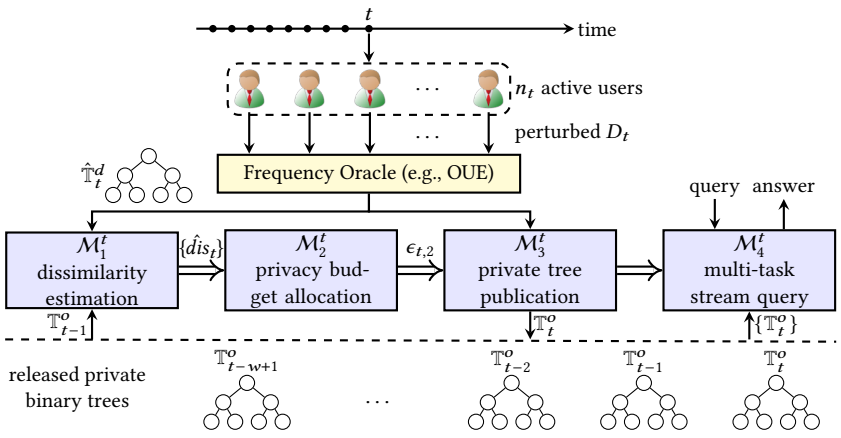


Fig. 2. The framework of MTSP-LDP

- **Private Dissimilarity Estimation** $\mathcal{M}_1^t$. At each time t, the server computes the dissimilarity dis_t between the statistics of current user data D_t and the last released stream statistics, and dis_t will be used to determine whether to approximate with the previous release or publish with noise. The main challenge we need to address in $\mathcal{M}_1^t$ is how to privately compute dis_t

as user data D_t is not available to the server in LDP, and instead, we design an unbiased estimator $\hat{dis}_t$ to estimate dis_t while preserving user privacy.

- **Optimal Privacy Budget Allocation** $\mathcal{M}_2^t$. Given $\hat{dis}_t$, existing methods [28, 37] directly compare $\hat{dis}_t$ with the publication error err_t. If $\hat{dis}_t < err_t$, then they approximate with the previous release; otherwise, they publish with noise using a predetermined privacy budget. However, we notice that this commonly used strategy is often not optimal, particularly in the w-event setting, as it bases its decision solely on the current timestamp while neglecting the temporal context within the sliding window. We therefore propose an optimal privacy budget allocation strategy by cleverly leveraging previously computed dissimilarities, i.e., $\{\hat{dis}_{t'}\}_{t' \le t}$. This strategy will determine whether to approximate or to publish, and how to optimally allocate the privacy budget at time t.
- **Private Adaptive Tree Publication** $\mathcal{M}_3^t$. In MTSP-LDP, the server privately collects user data D_t and represents D_t as a private binary tree to both protect user privacy and support multi-task streaming query. This private binary tree participates in dissimilarity estimation in $\mathcal{M}_1^t$ as well as streaming query in $\mathcal{M}_4^t$. We construct the private binary tree in a data-adaptive way in order to accurately capture the statistics of user data D_t. The output tree $\mathbb{T}_t^o$ is then released and will be used by $\mathcal{M}_4^t$ to run streaming queries.
- **Budget-Free Multi-task Streaming Query** $\mathcal{M}_4^t$. Unlike existing research, which typically focuses on ad-hoc queries, e.g., counting, histogram query, or frequency query, MTSP-LDP is designed to support multi-task streaming queries, including all queries defined in §2.5, by utilizing the released private binary trees $\{\mathbb{T}_{t'}^o\}_{t' \le t}$ without incurring any additional privacy budget. We theoretically show that MTSP-LDP satisfies w-event LDP.

These above sub-mechanisms work in tandem to achieve privacy-preserving data analysis in streaming environments, as illustrated in Algorithm 1. Next, we will introduce them in detail.

Algorithm 1: MTSP-LDP

Input: Data stream $\mathcal{D} = (D_1, D_2, \ldots)$, privacy budget ϵ, sliding window size w
Output: Respond to queries Q_t^c, Q_t^r or Q_t^e at each time t

foreach $t = 1, 2, \ldots$ **do**
  if $t \le w$ **then**
    $\mathbb{T}_t^o \leftarrow \mathrm{FO}(D_t, \epsilon/w)$; // run FO on D_t with budget ϵ/w
  else
    // 1. private dissimilarity estimation
    $\hat{\mathbb{T}}_t^d \leftarrow \mathrm{FO}(D_t, \epsilon/(2w))$;
    $\hat{dis}_t \leftarrow \mathrm{dissimilarity}(\hat{\mathbb{T}}_t^d, \mathbb{T}_{t-1}^o)$;
    // 2. optimal privacy budget allocation
    $\epsilon_{t,2} \leftarrow$ run $\mathcal{M}_3^t(\{\hat{dis}_t\})$;
    // 3. private tree publication
    if $\epsilon_{t,2} > 0$ **then**
      $\hat{\mathbb{T}}_t^a \leftarrow \mathtt{ATC}(\hat{\mathbb{T}}_t^d)$; // adaptive tree construction
    else
      $\hat{\mathbb{T}}_t^a \leftarrow \mathbb{T}_{t-1}^o$;
    $\mathbb{T}_t^o \leftarrow \mathtt{GroupSmooth}(\hat{\mathbb{T}}_t^a)$;
  // 4. multi-task processing
  Use $\{\mathbb{T}_{t'}^o\}_{t' \le t}$ to respond to queries Q_t^c, Q_t^r or Q_t^e.

3.2 Private Dissimilarity Estimation

We are now ready to describe each mechanism in MTSP-LDP. Recall that MTSP-LDP follows the dissimilarity guided publication framework, where the server checks at every timestamp whether it is more beneficial to approximate the current statistics with the last released statistics, than to publish new statistics with noise. This requires the server to compute the dissimilarity between user data D_t at time t and the last released statistics. It will be clear later that the last released statistics is also represented as a private binary tree, denoted by $\mathbb{T}^o_{t-1}$ (see §3.4). To compare the dissimilarity between D_t and $\mathbb{T}^o_{t-1}$, we can convert D_t to a private binary tree $\mathbb{T}^d_t$ using the method described in §2.4, and the dissimilarity is defined as the mean of the squared differences between corresponding node properties in these two trees, i.e.,

$$dis_t = \text{dissimilarity}(\mathbb{T}^d_t, \mathbb{T}^o_{t-1}) \triangleq \frac{1}{|\mathbb{T}^d_t|} \sum_{a \in \mathbb{T}^d_t} (\mathbb{T}^d_t[a] - \mathbb{T}^o_{t-1}[a])^2,$$

where $|\mathbb{T}^d_t|$ denotes the number of nodes in the tree.

However, in the LDP setting, the server cannot directly access user data D_t, hence $\mathbb{T}^d_t$ is unknown. Instead, the server can estimate $\mathbb{T}^d_t$ by $\hat{\mathbb{T}}^d_t$ using several FOs, as explained in §2.4. Next, we present an unbiased estimator of dis_t.

Theorem 1. *If $\hat{\mathbb{T}}^d_t$ is estimated with privacy budget $\epsilon_{t,1}$, then $\hat{dis}_t$ is an unbiased estimate of dis_t and satisfies $\epsilon_{t,1}$-LDP, where*

$$\hat{dis}_t \triangleq \frac{1}{|\hat{\mathbb{T}}^d_t|} \sum_{a \in \hat{\mathbb{T}}^d_t} (\hat{\mathbb{T}}^d_t[a] - \mathbb{T}^o_{t-1}[a])^2 - var(\epsilon_{t,1}).$$

Proof. We begin by computing the expectation of $\hat{dis}_t$:

$$\mathbb{E}(\hat{dis}_t) = \mathbb{E}\Big[\frac{1}{|\hat{\mathbb{T}}^d_t|} \sum_{a \in \hat{\mathbb{T}}^d_t} (\hat{\mathbb{T}}^d_t[a] - \mathbb{T}^o_{t-1}[a])^2 - var(\epsilon_{t,1})\Big]$$

Since $var(\epsilon_{t,1})$ is a constant, it can be factored out of the expectation:

$$\begin{aligned}
\mathbb{E}(\hat{dis}_t) &= \frac{1}{|\hat{\mathbb{T}}^d_t|} \sum_{a \in \hat{\mathbb{T}}^d_t} \mathbb{E}\Big[(\hat{\mathbb{T}}^d_t[a] - \mathbb{T}^o_{t-1}[a])^2\Big] - \mathbb{E}\Big[var(\epsilon_{t,1})\Big] \\
&= \frac{1}{|\hat{\mathbb{T}}^d_t|} \sum_{a \in \hat{\mathbb{T}}^d_t} \Big[(\mathbb{T}^d_t[a] - \mathbb{T}^o_{t-1}[a])^2 + var(\hat{\mathbb{T}}^d_t[a])\Big] - var(\epsilon_{t,1}) \\
&= \frac{1}{|\hat{\mathbb{T}}^d_t|} \sum_{a \in \hat{\mathbb{T}}^d_t} \Big[(\mathbb{T}^d_t[a] - \mathbb{T}^o_{t-1}[a])^2 + var(\hat{\mathbb{T}}^d_t[a]) - var(\epsilon_{t,1})\Big]
\end{aligned}$$

Since $\hat{\mathbb{T}}^d_t[a]$ is generated by the FO mechanism with budget $\epsilon_{t,1}$, we have $var(\hat{\mathbb{T}}^d_t[a]) = var(\epsilon_{t,1})$. Thus, the expectation becomes:

$$\mathbb{E}(\hat{dis}_t) = \frac{1}{|\hat{\mathbb{T}}^d_t|} \sum_{a \in \hat{\mathbb{T}}^d_t} \Big[(\mathbb{T}^d_t[a] - \mathbb{T}^o_{t-1}[a])^2\Big] = dis_t$$

Finally, because differential privacy is immune to post-processing [20], estimator $\hat{dis}_t$ is still $\epsilon_{t,1}$-LDP as long as $\hat{\mathbb{T}}^d_t$ is $\epsilon_{t,1}$-LDP. □

Remarks. The dissimilarity is estimated at every timestamp using a fixed privacy budget $\epsilon_{t,1} = \epsilon/(2w)$, i.e., half of the total privacy budget uniformly spent on each timestamp in the window. Since differential privacy is immune to post-processing [20], estimator $\hat{dis}_t$ is still $\epsilon_{t,1}$-LDP as long as $\hat{\mathbb{T}}_t^d$ is $\epsilon_{t,1}$-LDP.

3.3 Optimal Privacy Budget Allocation

The server now needs to make a decision at time t whether it should approximate the current statistics using the last released statistics (referred to as *approximation*), or publish the current statistics with the necessary noise (referred to as *publication*). Both options will introduce errors in the released statistics, i.e., the approximation incurs error $\hat{dis}_t$, and the publication incurs error $var(\epsilon_{t,2})$ which is the estimation error of running FO with privacy budget $\epsilon_{t,2}$.

Existing methods [28, 37] directly compare $\hat{dis}_t$ with $var(\epsilon_{t,2})$, and if $\hat{dis}_t < var(\epsilon_{t,2})$, then the server chooses approximation, otherwise it chooses publication. We notice that this commonly used strategy is myopic, focusing only on the current timestamp and neglecting long-term overall accuracy of the released statistics. Recent empirical studies also show that these existing methods often perform worse than even static baseline methods [40]. To address this weakness, we propose a novel strategy, i.e., ***O**ptimal privacy **B**udget **A**llocation (OBA)*, that aims to achieve high long-term accuracy of released statistics.

The idea of OBA is that, at every time t, we make a decision of either approximation or publication by considering the benefit in a long-term period of w timestamps backwards, rather than only the current timestamp. In more detail, as we already know the dissimilarities $\hat{dis}_{t-w+1}, \ldots, \hat{dis}_t$, an optimal privacy budget allocation strategy in a time window of size w should choose publication on those timestamps with large dissimilarities (say, top k largest dissimilarities), as choosing approximation on these k timestamps is only likely to introduce larger errors in the released statistics. Hence, a good strategy should spend the remaining $\epsilon/2$ privacy budget on these k timestamps. Once a timestamp is selected for publication, its approximation error $\hat{dis}_t$ is replaced by a publication error. The goal then is to minimize the total publication error (the sum of variances) across these k selected timestamps under a fixed total budget of $\epsilon/2$. Since the magnitude of the original $\hat{dis}_t$ no longer affects that timestamp's error and $var(\cdot)$ is convex in the budget, the total publication error is minimized by dividing the budget equally among all the k timestamps. The remaining $w - k$ timestamps with small dissimilarities simply choose approximation. Finally, if the current time t belongs to these k timestamps, then the server chooses publication with privacy budget $\epsilon_{t,2} = \epsilon/(2k)$; otherwise, the server chooses approximation. OBA is illustrated in Fig. 3.

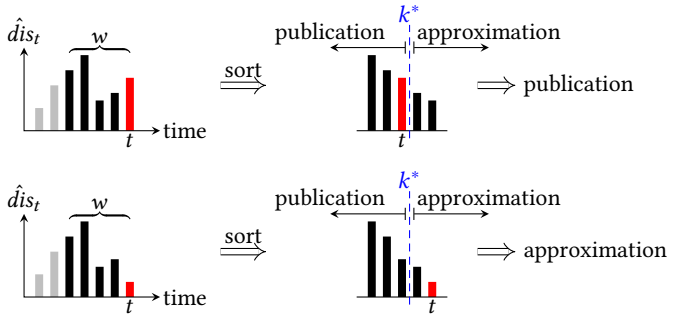


Fig. 3. Illustration of OBA

The last problem is how to find an optimal k. Let $\hat{dis}_{(1)}, \ldots, \hat{dis}_{(w)}$ denote the w dissimilarities $\hat{dis}_{t-w+1}, \ldots, \hat{dis}_t$ sorted in descending order. The *cumulative error* of the released statistics in a time window of size w consists of two parts: (i) cumulative publication error at these k timestamps due to running k FOs each using a privacy budget $\frac{\epsilon}{2k}$, thus incurring an error $k \cdot var(\frac{\epsilon}{2k})$, and (ii) the cumulative approximation error on the remaining $w-k$ timestamps, incurring another error $\sum_{i=k+1}^{w} \hat{dis}_{(i)}$. Furthermore, we should allow $k=0$, and in this case, the server always chooses approximation in the current window, thus incurring an error $\sum_{i=1}^{w} \hat{dis}_{(i)}$.

Let $E_t(k)$ denote the cumulative error of the released statistics in the most recent window of size w at time t, expressing k as a parameter. Then, we have

$$E_t(k) = \begin{cases} \sum_{i=1}^{w} \hat{dis}_{(i)}, & k = 0, \\ k \cdot var(\frac{\epsilon}{2k}) + \sum_{i=k+1}^{w} \hat{dis}_{(i)}, & 0 < k \leq w. \end{cases}$$

Our goal is to find an optimal k^* by minimizing $E_t(k)$, i.e.,

$$k^* \in \arg\min_{0 \leq k \leq w} E_t(k). \tag{1}$$

As k is in a finite set $\{0, \ldots, w\}$, a simple method to solve Problem (1) is to enumerate each k and choose one minimizing $E_t(k)$. The pseudo-code of OBA is given in Alg. 2.

Algorithm 2: **O**ptimal Privacy **B**udget **A**llocation (OBA)

Input: Dissimilarities $\hat{dis}_{t-w+1}, \ldots, \hat{dis}_t$, privacy budget ϵ, remaining privacy budget for current window ϵ_t^{rm}
Output: Optimal privacy budget $\epsilon_{t,2}$ for timestamp t

1 $[\hat{dis}_{(1)}, \ldots, \hat{dis}_{(w)}] \leftarrow \text{Sort}([\hat{dis}_{t-w+1}, \ldots, \hat{dis}_t])$; `// in descending order`
2 $[t_{(1)}, \ldots, t_{(w)}] \leftarrow$ timestamps of $[\hat{dis}_{(1)}, \ldots, \hat{dis}_{(w)}]$;
3 $E_t[0] \leftarrow \sum_{i=1}^{w} \hat{dis}_{(i)}$;
4 **for** $k \leftarrow 1$ **to** w **do** $E_t[k] \leftarrow k \cdot var(\frac{\epsilon}{2k}) + \sum_{i=k+1}^{w} \hat{dis}_{(i)}$;
5 $k^* \leftarrow \arg\min_k E_t[k]$;
6 **if** $t \in \{t_{(1)}, \ldots, t_{(k^*)}\}$ **then** `// if publication`
7 $\quad \epsilon_{t,2} \leftarrow \min(\epsilon_t^{rm}, \frac{\epsilon}{2k^*})$; `// publication using budget` $\epsilon_{t,2}$
8 **else** $\epsilon_{t,2} \leftarrow 0$; `// approximation using previous time step`
9 **return** $\epsilon_{t,2}$;

It is noteworthy to mention that the ultimate goal of OBA is to decide whether we choose approximation or publication at time t, and if it is publication, how much privacy budget to spend. In Line 6, if current timestamp t belongs to the selected k^* timestamps, then we choose publication using privacy budget $\epsilon_{t,2}$; otherwise we choose approximation with no privacy budget (Line 8). In addition, $\epsilon_{t,2}$ should be upper bounded by the remaining privacy budget ϵ_t^{rm} available in current window (Line 7).

Improving Efficiency. A straightforward implementation of OBA requires sorting dissimilarities and enumerating all possible k, resulting in a per-timestamp time complexity of $O(w \log w + w^2) = O(w^2)$. When w is large, such brute-force enumeration becomes computationally expensive in streaming settings. We present several optimizations to improve its computational efficiency.

- Incremental Sorting. Instead of sorting dissimilarities from scratch (Line 1) at each timestamp, we maintain the sorted sequence $\{\hat{dis}_{(1)}, \ldots, \hat{dis}_{(w)}\}$ incrementally. By using a tree-based

data structure, the insertion of the new $\hat{dis}_t$ and the deletion of the expired $\hat{dis}_{t-w}$ can both be performed in $O(\log w)$ time.

- Constant-Time Error Evaluation. The computation of the cumulative error $E_t(k)$ (Line 4) can be optimized by reusing intermediate results. The publication error term $k \cdot var(\frac{\epsilon}{2k})$ is time invariant which depends only on k, thus it can be computed offline and stored. The approximation error term can be computed incrementally while sorting (and similarly for the sum in Line 3). So calculating E_t actually has time complexity $O(w)$.
- Early Termination. The function $E_t(k)$ typically exhibits a unimodal structure (initially decreasing and then increasing) or monotonicity. Specifically, $E_t(k)$ decreases when the benefit of removing a large approximation error $\hat{dis}_{(k+1)}$ outweighs the cost of added noise, and begins to increase when the remaining dissimilarities are small. This structure enables an early-stop strategy: we iterate k starting from 0 and terminate immediately once $E_t(k+1) > E_t(k)$. Let k_{stop} denote the number of values evaluated before termination. Thus, we only need to evaluate k_{stop} candidates instead of all w options.

By combining these techniques, the overall time complexity is reduced to $O(\log w + k_{stop})$, ensuring scalability even with large window sizes.

3.4 Private Adaptive Tree Publication

In the previous mechanism, if the output privacy budget $\epsilon_{t,2} > 0$, then the server will choose publication with error using privacy budget $\epsilon_{t,2}$ to release current stream statistics, which are represented as a private binary tree, denoted by $\mathbb{T}_t^o$. We now describe how to build tree $\mathbb{T}_t^o$ using privacy budget $\epsilon_{t,2}$.

A straightforward way to build tree $\mathbb{T}_t^o$ is to use the method we introduced in §2.4. That is, we build $\mathbb{T}_t^o$ by invoking an FO at each layer with privacy budget $\epsilon_{t,2}$. The issue of this approach is that, user data D_t in practice may be not evenly distributed. There are few "hot" values (e.g., few popular places many people visit), and many "cold" values (e.g., many places people rarely visit) in D_t. In this case, nodes in $\mathbb{T}_t^o$ representing cold values will have very small property values (i.e., frequencies), and after adding noise by FO, their estimation accuracy is likely to diminish, resulting in little to no utility.

To address these issues, a data-adaptive hierarchical structure is commonly employed for static data. Such a structure is typically constructed level by level: at each level, a central server interacts with users to obtain interval frequency counts and decides whether to further partition those intervals. Although this approach incurs latency that grows with tree height and is impractical in streaming scenarios, it is acceptable in static scenarios, where the private tree needs to be built only once. However, efficiently addressing this challenge in streaming data, where low latency is required, remains an open problem.

A key novelty of MTSP-LDP lies in its effective use of the intermediate tree $\hat{\mathbb{T}}_t^d$ from $\mathcal{M}_1^t$, which has been largely overlooked in prior methods. We propose a *Data-**A**daptive Private Binary **T**ree **C**onstruction (ATC)* method that builds a private binary tree $\hat{\mathbb{T}}_t^a$ able to better capture the statistics of user data D_t than the straightforward method. Furthermore, we propose a *grouping and smoothing* strategy to refine trees $\{\hat{\mathbb{T}}_{t'}^a\}_{t' \le t}$, which can further improve the estimation accuracy of the final released tree $\mathbb{T}_t^o$.

3.4.1 Data-Adaptive Private Binary Tree Construction (ATC). The idea of ATC is to leverage the tree $\hat{\mathbb{T}}_t^d$ as an auxiliary tree to build a data-adaptive private binary tree $\hat{\mathbb{T}}_t^a$ to better capture the statistics of user data D_t than building the tree from scratch. Recall that $\hat{\mathbb{T}}_t^d$ is also built from D_t but using privacy budget $\epsilon_{t,1} = \epsilon/(2w)$ at time t. Hence, the tree $\hat{\mathbb{T}}_t^d$ also contains information about D_t. We

propose to use the node properties stored in $\hat{\mathbb{T}}_t^d$ to decide whether we need to prune the tree, i.e., remove branches representing cold values. This approach allows the entire adaptive tree structure to be constructed at once, without consuming additional privacy budget or requiring extra rounds of user interaction. Then, FO is only applied to the pruned tree to estimate node properties. Finally, node properties in removed branches are directly inferred from their parents.

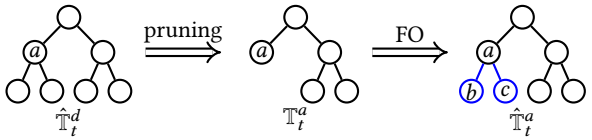


Fig. 4. Illustration of ATC

We use the example in Fig. 4 to further explain the above idea. Given $\hat{\mathbb{T}}_t^d$ from $\mathcal{M}_1^t$, we check the node property from top to bottom, layer by layer. If some node a's property $\hat{\mathbb{T}}_t^d[a] < \vartheta_1$, where ϑ_1 denotes a threshold frequency, then we prune the subtree rooted at node a while only keeping node a. This results a pruned tree $\mathbb{T}_t^a$. Then, we run two FOs to estimate the bottom two layers' node properties (cf. §2.4), and obtain a tree $\hat{\mathbb{T}}_t^a$. For pruned nodes, their properties are directly inferred from their parents, e.g., $\hat{\mathbb{T}}_t^a[b] = \hat{\mathbb{T}}_t^a[c] = \hat{\mathbb{T}}_t^a[a]/2$.

Algorithm 3: Data-**A**daptive Private Binary **T**ree **C**onstruction (ATC)

Input: Private binary tree $\hat{\mathbb{T}}_t^d$, threshold ϑ_1, privacy budget $\epsilon_{t,2}$
Output: Private binary tree $\hat{\mathbb{T}}_t^a$
`// Build tree` $\mathbb{T}_t^a$ `by pruning tree` $\hat{\mathbb{T}}_t^d$
1 Initialize an empty tree $\mathbb{T}_t^a$ with only root node r;
2 $current\_layer \leftarrow \{r\}$, $level \leftarrow 0$, $h \leftarrow$ height of tree $\hat{\mathbb{T}}_t^d$;
3 **while** $current\_layer \neq \emptyset$ *and* $level < h$ **do**
4 $next\_layer \leftarrow \emptyset$;
5 **foreach** *node* $a \in current\_layer$ **do**
6 **if** $\hat{\mathbb{T}}_t^d[a] \geq \vartheta_1$ **then**
7 Equally split node a into two child nodes c_1, c_2;
8 $next\_layer \leftarrow next\_layer \cup \{c_1, c_2\}$;
9 $level \leftarrow level + 1$;
10 $current\_layer \leftarrow next\_layer$;
`// Estimate each node's property using an FO (cf. §2.4)`
11 $\hat{\mathbb{T}}_t^a \leftarrow \mathrm{FO}(\mathbb{T}_t^a, \epsilon_{t,2})$;
`// Infer nodes' properties in the pruned tree`
12 **foreach** *node* $a \in \hat{\mathbb{T}}_t^a$ **do**
13 **if** *node a has no child and its depth* $< h$ **then**
14 Crete two child nodes c_1, c_2 for node a;
15 $\hat{\mathbb{T}}_t^a[c_1] \leftarrow \hat{\mathbb{T}}_t^a[a]/2$, $\hat{\mathbb{T}}_t^a[c_2] \leftarrow \hat{\mathbb{T}}_t^a[a]/2$;
16 **return** $\hat{\mathbb{T}}_t^a$;

The pseudo-code of ATC is given in Alg. 3. We first create a pruned tree $\mathbb{T}_t^a$ from the tree $\hat{\mathbb{T}}_t^d$ (Lines 1 to 10). Then we invoke an FO to estimate each node's property and obtain the tree $\hat{\mathbb{T}}_t^a$

(Line 11). Finally, pruned nodes' properties are directly inferred from their parents (Lines 12 to 15). It is worth noting that the resulting tree $\hat{\mathbb{T}}_t^a$ is still a perfect binary tree.

3.4.2 Grouping and Smoothing. The data-adaptive private binary tree $\hat{\mathbb{T}}_t^a$ captures the instantaneous statistics of the stream at time t, which may change a lot over time if the stream is highly variable. Instead of releasing $\hat{\mathbb{T}}_t^a$ as the stream statistics, we propose to release steady statistics, which can capture the underlying trend of the stream. Queries on the steady statistics will be more accurate and meaningful than on the instantaneous statistics.

To this end, inspired by Pegasus [11], we introduce a *grouping and smoothing* post-processing module. Different from Pegasus which is designed for CDP and user data is available to the server, we design grouping and smoothing module in the w-event LDP setting without directly accessing user data. The grouping and smoothing module processes trees $\{\hat{\mathbb{T}}_{t'}^a\}_{t'\le t}$, and outputs steady statistics $\{\mathbb{T}_{t'}^o\}_{t'\le t}$, which are also private binary trees.

Grouping. For trees $\{\hat{\mathbb{T}}_{t'}^a\}_{t'\le t}$, we consider a same node a at different time in these trees. We want to group the most recent similar node properties of node a into a group, and use the aggregate statistics of node property in this group as node a's property in the released tree $\mathbb{T}_t^o$. Specifically, let $T_a = \{t-l, \ldots, t-1\}$ denote a set of l timestamps at which node a's properties are similar and belong to a group at time $t-1$. At time t, we need to determine whether t belongs to group T_a or not. Recall that $\hat{\mathbb{T}}_t^a[a]$ is the estimated node property of a at time t, and we let $\mathbb{T}_t^a[a]$ denote its true property value. To measure the deviation of $\mathbb{T}_t^a[a]$ to the group, we define the *squared deviation* of node a by

$$\sigma_a^2 \triangleq (\mathbb{T}_t^a[a] - \frac{1}{l}\sum_{i=1}^{l} \mathbb{T}_{t-i}^a[a])^2,$$

and if $\sigma_a^2 \le \vartheta_2$ for some threshold ϑ_2, we add t to T_a; otherwise node a's group becomes to $T_a = \{t\}$ at time t. The challenge of the grouping operation is that the true node properties $\{\mathbb{T}_{t'}^a[a]\}_{t'\le t}$ are unknown in the LDP setting, and we only know their estimates $\{\hat{\mathbb{T}}_{t'}^a[a]\}_{t'\le t}$. We provide an unbiased estimator of σ_a^2.

Theorem 2. *Estimator $\hat{\sigma}_a^2$ is an unbiased estimate of σ_a^2, i.e.,*

$$\hat{\sigma}_a^2 = \big(\hat{\mathbb{T}}_t^a[a] - \frac{1}{l}\sum_{i=1}^{l} \hat{\mathbb{T}}_{t-i}^a[a]\big)^2 - \frac{l+1}{l} var(\epsilon_{t,2}).$$

Proof. First, compute the expectation of $\hat{\sigma}_a^2 + \frac{l+1}{l} var(\epsilon_{t,2})$:

$$\begin{aligned}
&\mathbb{E}\big(\hat{\sigma}_a^2 + \frac{l+1}{l} var(\epsilon_{t,2})\big) \\
&= \mathbb{E}\Big[\big(\hat{\mathbb{T}}_t^a[a] - \frac{1}{l}\sum_{i=1}^{l} \hat{\mathbb{T}}_{t-i}^a[a]\big)^2\Big] \\
&= \mathbb{E}\Big[\hat{\mathbb{T}}_t^a[a]^2 - \frac{2}{l}\hat{\mathbb{T}}_t^a[a]\sum_{i=1}^{l} \hat{\mathbb{T}}_{t-i}^a[a] + (\frac{1}{l}\sum_{i=1}^{l} \hat{\mathbb{T}}_{t-i}^a[a])^2\Big] \\
&= \mathbb{E}(\hat{\mathbb{T}}_t^a[a]^2) - \frac{2}{l}\mathbb{T}_t^a[a]\sum_{i=1}^{l} \mathbb{T}_{t-i}^a[a] + \mathbb{E}\Big[(\frac{1}{l}\sum_{i=1}^{l} \hat{\mathbb{T}}_{t-i}^a[a])^2\Big] \\
&= var(\hat{\mathbb{T}}_t^a[a]) + \mathbb{T}_t^a[a]^2 - \frac{2}{l}\mathbb{T}_t^a[a]\sum_{i=1}^{l} \mathbb{T}_{t-i}^a[a]
\end{aligned}$$

$$+ var\Big(\frac{1}{l}\sum_{i=1}^{l}\hat{\mathbb{T}}^a_{t-i}[a]\Big) + \Big(\frac{1}{l}\sum_{i=1}^{l}\mathbb{T}^a_{t-i}[a]\Big)^2$$

$$= \Big(\mathbb{T}^a_t[a] - \frac{1}{l}\sum_{i=1}^{l}\mathbb{T}^a_{t-i}[a]\Big)^2 + var(\hat{\mathbb{T}}^a_t[a]) + var\Big(\frac{1}{l}\sum_{i=1}^{l}\hat{\mathbb{T}}^a_{t-i}[a]\Big)$$

$$= \sigma_a^2 + var(\hat{\mathbb{T}}^a_t[a]) + var\Big(\frac{1}{l}\sum_{i=1}^{l}\hat{\mathbb{T}}^a_{t-i}[a]\Big)$$

Notice that

$$var(\hat{\mathbb{T}}^a_t[a]) = var(\epsilon_{t,2})$$

and

$$var\Big(\frac{1}{l}\sum_{i=1}^{l}\hat{\mathbb{T}}^a_{t-i}[a]\Big) = \frac{1}{l^2}\sum_{i=1}^{l}var(\hat{\mathbb{T}}^a_{t-i}[a]) = \frac{1}{l}var(\epsilon_{t,2})$$

Therefore, we conclude that

$$\mathbb{E}(\hat{\sigma}_a^2) = \sigma_a^2.$$

This completes the proof. □

Smoothing. At each time t, we now know every node a belongs to a most recent group T_a with similar node properties at different time. To release the statistics with regard to node a, we propose to compute the aggregate statistics of node properties in the group, e.g., mean, median, etc. Hence, we define

$$\mathbb{T}^o_t[a] \triangleq \text{Aggregate}(\{\hat{\mathbb{T}}^a_{t'}[a] : t' \in T_a\})$$

and release tree $\mathbb{T}^o_t$ as the stream statistics at time t. Querying on $\mathbb{T}^o_t$ will provide more accurate and meaningful results than on $\hat{\mathbb{T}}^a_t$.

Remarks. It is noteworthy to mention that grouping and smoothing are post-processing operations, both of which operate on perturbed statistics without consuming additional privacy budgets.

3.5 Budget-Free Multi-Task Streaming Query

Based on the released stream statistics $\{\mathbb{T}^o_{t'}\}_{t'\le t}$, we can process multiple streaming query tasks defined in §2.5.

3.5.1 Counting Query. To answer a counting query $Q^c_t(v,\Delta)$ at time t, we traverse each tree $\mathbb{T}^o_{t'}$ such that $t - t' < \Delta$, and retrieve the property of the leaf node a_v representing private value v, and hence

$$Q^c_t(v,\Delta) = \sum_{t'=t-\Delta+1}^{t}\mathbb{T}^o_{t'}[a_v]n_{t'}.$$

3.5.2 Range Query. To answer a range query $Q^r_t([v_1,v_2],\Delta)$ at time t, we find a minimum cover in the released tree such that they jointly cover the query range $[v_1,v_2]$ (cf. Fig. 1). Denote the minimum cover by $c(v_1,v_2)$, then

$$Q^r_t([v_1,v_2],\Delta) = \sum_{t'=t-\Delta+1}^{t}\sum_{a\in c(v_1,v_2)}\mathbb{T}^o_{t'}[a]n_{t'}.$$

3.5.3 Event Monitoring. Because an event monitoring query $Q^e_t(\alpha,\beta)$ actually consists of several counting or range queries, it can be efficiently answered as above.

4 Privacy Analysis and Parameter Selection

4.1 Privacy Analysis

We have the following privacy guarantee for MTSP-LDP.

Theorem 3. *MTSP-LDP satisfies w-event ϵ-LDP for each user.*

Proof. MTSP-LDP ensures that the total privacy budget consumed within any sliding window of length w does not exceed ϵ. Specifically, at each timestamp t, the budget ϵ is partitioned into two parts, i.e., $\epsilon/2$ is evenly divided among the w mechanisms $\mathcal{M}_1^{t'}$ where $t' \in [t-w+1, t]$, and the remaining $\epsilon/2$ is dynamically distributed among the corresponding w mechanisms $\mathcal{M}_2^{t'}$. As explained in §3.3, the allocation strategy guarantees that the cumulative budget consumed by all $\mathcal{M}_2^{t'}$ over any sliding window of length w does not exceed $\epsilon/2$. Mechanisms $\mathcal{M}_3^t$ and $\mathcal{M}_4^t$ only process perturbed data and do not access user data, i.e., they are post-processing operations, thus incurring no additional privacy cost. Therefore, the total privacy budget spent within any sliding window is bounded by ϵ, and MTSP-LDP satisfies w-event ϵ-LDP. □

4.2 Selection of Thresholds ϑ_1 and ϑ_2

Two important parameters in MTSP-LDP are thresholds ϑ_1 and ϑ_2 used in §3.4. We discuss how to choose these thresholds to maximize the overall utility of our framework.

4.2.1 Selection of Threshold ϑ_1. Threshold ϑ_1 is used for pruning tree $\hat{\mathbb{T}}_t^d$ in Alg. 3 to obtain a data-adaptive tree $\hat{\mathbb{T}}_t^a$ after invoking FOs for each layer. We find that invoking FOs on the pruned tree may reduce estimation error if threshold ϑ_1 is carefully chosen. To understand the reason, without loss of generality, let us consider a tree without pruning and a tree pruned at node a, respectively (see Fig. 5). If we spend the same amount of privacy budget $\epsilon_{t,2}$ and use the same number of users to estimate the node frequencies in each layer of these two trees, then the estimate of each node frequency will have the same variance $var(\epsilon_{t,2})$. While for the pruned tree, properties of those pruned nodes are approximated by recursively halving their parent nodes' properties. Therefore, estimation errors are different only for the two subtrees rooted at node a, excluding node a. Assume the subtree has height $h+1$.

For the left tree, the total estimation error of the subtree rooted at node a (excluding node a) is

$$err_1 \triangleq \sum_{i=1}^{h} 2^i \mathbb{E}[(\hat{f}_i - f_i)^2] = 2(2^h - 1)\, var(\epsilon_{t,2}).$$

For the right tree, the total estimation error of the subtree rooted at node a (excluding node a) is

$$\begin{aligned}
err_2 &\triangleq \sum_{i=1}^{h} 2^i \mathbb{E}[(\hat{f}_i - f_i)^2] = \sum_{i=1}^{h} 2^i \mathbb{E}[(\frac{\hat{f}_0}{2^i} - f_i)^2] \\
&= \sum_{i=1}^{h} 2^i \mathbb{E}(\frac{\hat{f}_0^2}{2^{2i}} - \frac{\hat{f}_0 f_i}{2^{i-1}} + f_i^2) = \sum_{i=1}^{h} 2^i (\frac{var(\epsilon_{t,2}) + f_0^2}{2^{2i}} - \frac{f_0 f_i}{2^{i-1}} + f_i^2) \\
&\le \sum_{i=1}^{h} 2^i (\frac{var(\epsilon_{t,2}) + \vartheta_1^2}{2^{2i}} + \vartheta_1^2) = \sum_{i=1}^{h} (\frac{var(\epsilon_{t,2}) + \vartheta_1^2}{2^i} + 2^i \vartheta_1^2) \\
&\le var(\epsilon_{t,2}) + \vartheta_1^2 + 2(2^h - 1)\vartheta_1^2 \\
&= var(\epsilon_{t,2}) + (2^{h+1} - 1)\vartheta_1^2.
\end{aligned}$$

If we require the estimation error for the right tree is no larger than the left tree, i.e., $err_2 \leq err_1$, then we obtain

$$\vartheta_1 \leq \sqrt{\left(\frac{2^{h+1}-3}{2^{h+1}-1}\right) \cdot var(\epsilon_{t,2})} \leq \sqrt{var(\epsilon_{t,2})}. \tag{2}$$

Therefore, if threshold ϑ_1 is chosen according to Condition (2), the pruned tree will have smaller overall estimation error than the full binary tree, thus demonstrating the usefulness of tree pruning operation in Alg. 3.

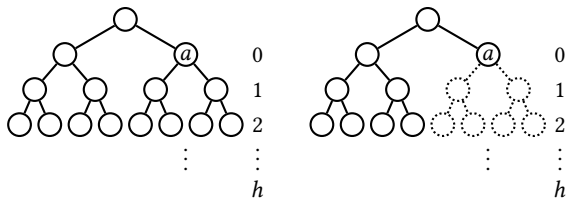


Fig. 5. Example of a same tree without pruning (left) and pruning at node a (right)

4.2.2 Selection of Threshold ϑ_2. The threshold ϑ_2 is used in the grouping and smoothing module (cf. §3.4.2). We find that the proper choice of ϑ_2 can also reduce estimation error.

To understand the reason, let us focus on an arbitrary node a in the tree. Let $T_a = \{t-l, \ldots, t\}$ denote a group of $l+1$ timestamps at which node a's properties are close. Let $g_i \triangleq \mathbb{T}^a_{t-i}[a]$ denote the true property value and $\hat{g}_i \triangleq \hat{\mathbb{T}}^a_{t-i}[a]$ denote its estimate. Let $\bar{g}_0 = \frac{1}{l+1}\sum_{i=0}^{l} \hat{g}_i$ denote the smoothed property estimate at time t (i.e., assume the aggregation function is the mean).

Without the grouping and smoothing operation, the estimation variance at time t is

$$err_3 \triangleq \mathbb{E}[(\hat{g}_0 - g_0)^2] = var(\epsilon_{t,2}).$$

In contrast, if we apply the grouping and smoothing operation, the estimation variance becomes

$$\begin{aligned}
err_4 \triangleq \mathbb{E}[(\bar{g}_0 - g_0)^2] &= \mathbb{E}\Big[\Big(\frac{1}{l+1}\sum_{i=0}^{l} \hat{g}_i - g_0\Big)^2\Big] \\
&= \mathbb{E}\Big[\Big(\frac{1}{l+1}\sum_{i=0}^{l} \hat{g}_i\Big)^2 - \frac{2}{l+1} g_0 \sum_{i=0}^{l} \hat{g}_i + g_0^2\Big] \\
&= var\Big(\frac{1}{l+1}\sum_{i=0}^{l} \hat{g}_i\Big) + \Big(\frac{1}{l+1}\sum_{i=0}^{l} g_i - g_0\Big)^2 \\
&= \Big(\frac{1}{l+1}\Big)^2 \sum_{i=0}^{l} var(\epsilon_{t-i,2}) + \Big[\frac{l}{l+1}\Big(\frac{1}{l}\sum_{i=1}^{l} g_i - g_0\Big)\Big]^2 \\
&= \Big(\frac{1}{l+1}\Big)^2 \sum_{i=0}^{l} var(\epsilon_{t-i,2}) + \Big(\frac{l}{l+1}\Big)^2 \sigma_a^2.
\end{aligned}$$

We require $err_4 \leq err_3$, implying

$$\sigma_a^2 \leq \frac{1}{l^2}\Big[(l+1)^2 var(\epsilon_{t,2}) - \sum_{i=0}^{l} var(\epsilon_{t-i,2})\Big].$$

Therefore, if we maintain the group for node a as long as the following condition holds, i.e.,

$$\vartheta_2 \leq \frac{1}{l^2}\Big[(l+1)^2 var(\epsilon_{t,2}) - \sum_{i=0}^{l} var(\epsilon_{t-i,2})\Big], \tag{3}$$

then the grouping and smoothing operation can reduce estimation error in our framework.

5 Evaluation

In this section, we evaluate the performance of MTSP-LDP on real-world datasets.

5.1 Datasets

We use four publicly available datasets, and we briefly describe them below.

- Cosmetics[1] is a seven-month dataset (Oct. 2019 to Apr. 2020) from a large e-commerce platform. We extract the last viewed item of each user per day, and this forms a value set of size 329. The stream consists of 1, 654, 771 users with 5, 112 timestamps.
- Taxi[2] is a collection of NYC yellow taxi trips from Jan. to Sep. 2024. Fare values are aggregated on a daily basis, and values above the 99.9-th percentile are removed. This yields a value set of size 150. The stream consists of 139, 713 users with 275 timestamps.
- Loan[3] is a collection of Lending Club loan records from 2007 to 2018. We aggregate loan amounts by issuance month and round values downward. This results in a value set of size 1, 719. The stream consists of 61, 992 users with 138 timestamps.
- Foursquare[4] is a collection of check-ins collected from a location-based social network Foursquare from Apr. 2012 to Sep. 2013. The value set is a set of cities of size 415. We count daily city-level check-ins and obtain a stream of 266, 909 users over 447 timestamps.

The statistics of these datasets are summarized in Table 1.

Table 1. Dataset statistics

dataset	stream length	domain size d	# of users n
Cosmetics	5, 112	329	1, 654, 771
Taxi	275	150	139, 713
Loan	138	1, 719	61, 992
Foursquare	447	415	266, 909

5.2 Settings

5.2.1 Metrics. To quantify the performance of the algorithm, we use different metrics tailored to each query task. Let $\hat{\theta}$ denote an estimator of the true value θ, and let $\hat{\theta}_i$ denote the estimate of the true value θ_i in the i-th query, for $i = 1, \ldots, n_q$.

- Mean Absolute Error (MAE). MAE quantifies the average magnitude of estimation errors and serves as a standard indicator of overall accuracy across all query types. The MAE of an

[1]https://www.kaggle.com/datasets/mkechinov/ecommerce-events-history-in-cosmetics-shop
[2]https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page
[3]https://www.kaggle.com/datasets/wordsforthewise/lending-club
[4]https://sites.google.com/site/yangdingqi/home/foursquare-dataset

estimator $\hat{\theta}$ is defined as

$$\mathsf{MAE}(\hat{\theta}) \triangleq \mathbb{E}(|\hat{\theta} - \theta|) = \frac{1}{n_q} \sum_{i=1}^{n_q} |\hat{\theta}_i - \theta_i|.$$

- Mean Relative Error (MRE). MRE measures the error relative to the true value, offering a scale-invariant perspective on accuracy. The MRE of an estimator $\hat{\theta}$ is defined as

$$\mathsf{MRE}(\hat{\theta}) \triangleq \mathbb{E}(\frac{|\hat{\theta} - \theta|}{\theta}) = \frac{1}{n_q} \sum_{i=1}^{n_q} \frac{|\hat{\theta}_i - \theta_i|}{\theta_i}.$$

- Receiver Operating Characteristic (ROC) Curve. For event monitoring, we use the ROC curve to evaluate the trade-off between true positive and false positive rates across varying decision thresholds. The Area Under the Curve (AUC) is used as a key metric for evaluating the model's overall performance. A higher AUC indicates better event detection performance.

5.2.2 Baselines. We compare MTSP-LDP with several state-of-the-art w-event LDP algorithms, including LBA, LBD, LSP, and LBU, as introduced in §2.6. To ensure fair comparison and reproducibility, all methods, including MTSP-LDP, were implemented in Python under a unified experimental framework. All experiments were conducted on a desktop equipped with an Intel Core i7-10700 CPU (2.90 GHz) and 16 GB of RAM.

5.3 Results

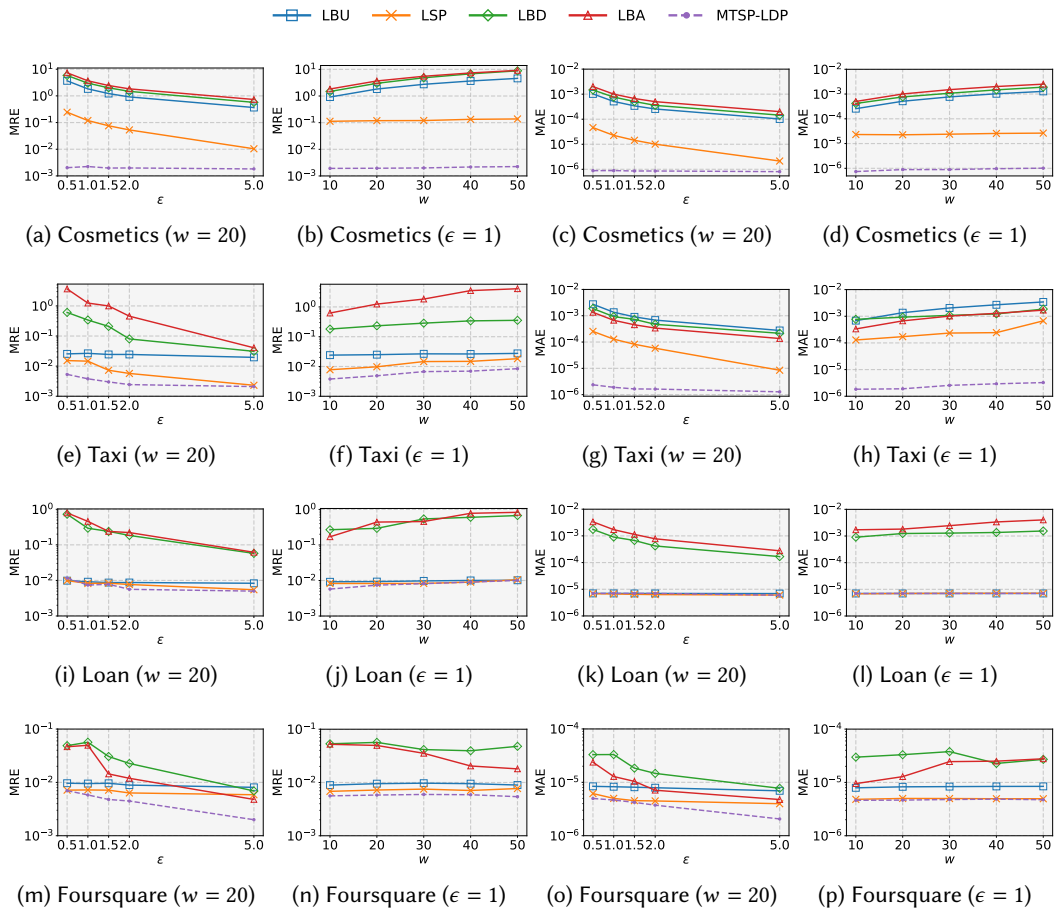
5.3.1 Evaluation for Counting Queries. In order to compare MTSP-LDP fairly with other baselines which are mainly designed for frequency histogram estimation, we let $\hat{\theta}_{t,v} \triangleq Q_t^c(v,1)/n_t$, and hence $\{\hat{\theta}_{t,v}\}_{v\in\Omega}$ is an estimate of the frequency histogram of the stream at time t. Then we evaluate the MAE and MRE of $\hat{\theta}_{v,t}$, averaged over time, and show the results on the four datasets in Fig. 6.

We observe that, for a fixed sliding window size w, as the privacy budget ϵ increases from 0.5 (i.e., high privacy protection) to 5 (i.e., low privacy protection), the estimation error for all algorithms decreases. Similarly, for a fixed privacy budget ϵ, as the window size w increases from 10 to 50, the estimation error increases accordingly. These results highlight a clear trade-off between privacy and data utility, where larger privacy budgets or smaller window sizes generally lead to better utility.

Notably, MTSP-LDP delivers consistently high accuracy across diverse datasets. As discussed in §3, this can be attributed to the fact that existing algorithms do not fully leverage the temporal correlations in the data stream when allocating privacy budgets. Although the LBD and LBA methods conserve privacy budgets during periods of minimal data change, their strategies focus solely on the current timestamp without considering the data distribution across the entire time window. This limitation is particularly critical in the w-event setting, where capturing the overall data distribution is crucial for accurate estimates. Besides, MTSP-LDP demonstrates a particularly significant advantage when the privacy budget is small, as smaller budgets inherently introduce larger errors, amplifying the weaknesses of other methods. This makes MTSP-LDP highly effective in scenarios requiring strict privacy guarantees.

5.3.2 Evaluation for Range Queries. For baseline methods with frequency histogram estimates $\{\hat{f}_{t,v}\}_{v\in\Omega}$, we can obtain the estimate of a frequency range query by

$$\hat{f}_{t,[v_1,v_2]} \triangleq \sum_{t'=t-\Delta+1}^{t} \sum_{v\in[v_1,v_2]} \hat{f}_{t',v}.$$



(a) Cosmetics ($w = 20$) (b) Cosmetics ($\epsilon = 1$) (c) Cosmetics ($w = 20$) (d) Cosmetics ($\epsilon = 1$)

(e) Taxi ($w = 20$) (f) Taxi ($\epsilon = 1$) (g) Taxi ($w = 20$) (h) Taxi ($\epsilon = 1$)

(i) Loan ($w = 20$) (j) Loan ($\epsilon = 1$) (k) Loan ($w = 20$) (l) Loan ($\epsilon = 1$)

(m) Foursquare ($w = 20$) (n) Foursquare ($\epsilon = 1$) (o) Foursquare ($w = 20$) (p) Foursquare ($\epsilon = 1$)

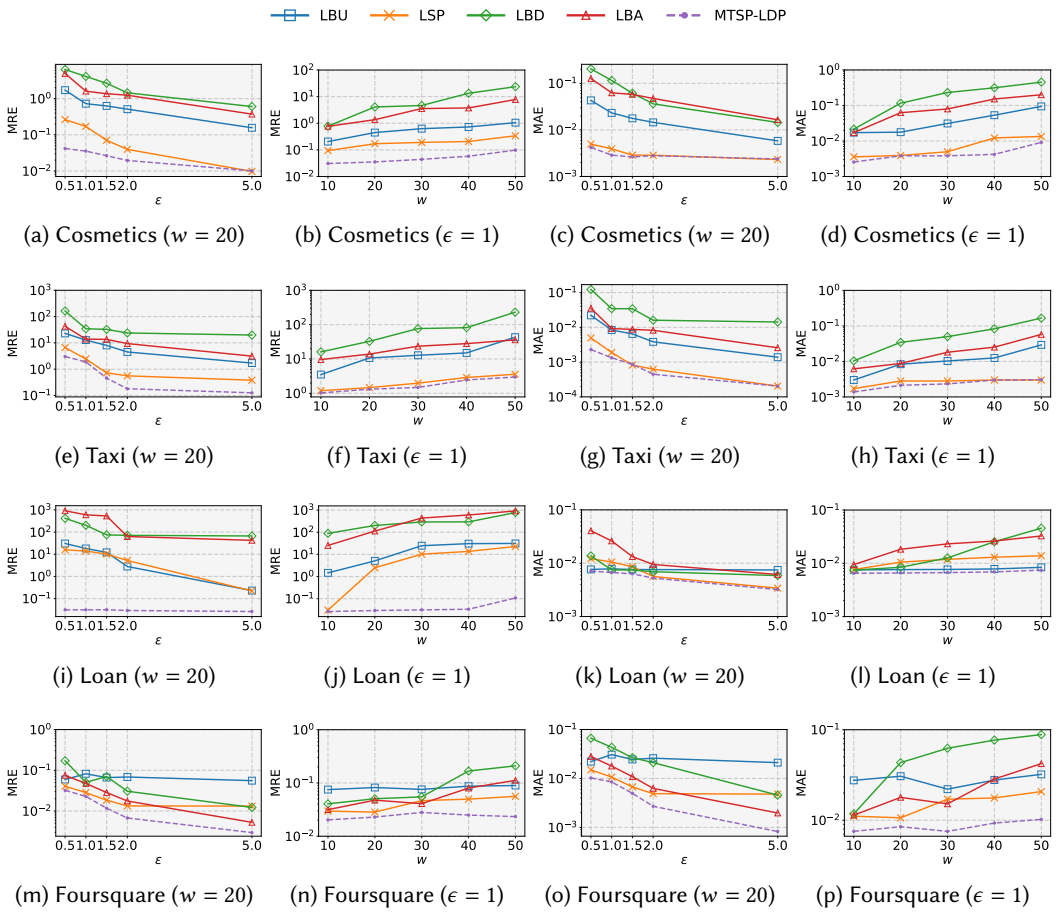
Fig. 6. Counting query evaluation with varying ϵ and w

For MTSP-LDP, the corresponding estimate is

$$\hat{\theta}_{t,[v_1,v_2]} \triangleq \frac{Q_t^r([v_1,v_2],\Delta)}{\sum_{t'=t-\Delta+1}^{t} n_{t'}}.$$

We randomly generate 50 range query tasks with varying value ranges $[v_1, v_2]$ and time spans Δ. For each query, the MRE and MAE between the estimates and ground truths are computed, and the average MRE and MAE across all queries are reported. The results are shown in Fig. 7. Note that some query tasks span time ranges exceeding the predefined window size, which means the total privacy budget used by these queries exceeds the set budget for a single window. However, this condition applies equally to all methods, ensuring a fair comparison.

We observe that MTSP-LDP consistently achieves the lowest estimation errors across all settings, with particularly pronounced improvements in MRE, demonstrating superior accuracy under normalized error metrics. This is because other approaches do not account for the complexity of performing complex query tasks on streaming data and instead rely solely on publishing frequency histograms. While frequency histograms can be used to derive results for complex queries, their cumulative error grows proportionally with the query range, leading to a sharp decline in data utility.



(a) Cosmetics ($w = 20$) (b) Cosmetics ($\epsilon = 1$) (c) Cosmetics ($w = 20$) (d) Cosmetics ($\epsilon = 1$)

(e) Taxi ($w = 20$) (f) Taxi ($\epsilon = 1$) (g) Taxi ($w = 20$) (h) Taxi ($\epsilon = 1$)

(i) Loan ($w = 20$) (j) Loan ($\epsilon = 1$) (k) Loan ($w = 20$) (l) Loan ($\epsilon = 1$)

(m) Foursquare ($w = 20$) (n) Foursquare ($\epsilon = 1$) (o) Foursquare ($w = 20$) (p) Foursquare ($\epsilon = 1$)

Fig. 7. Range query evaluation with varying ϵ and w

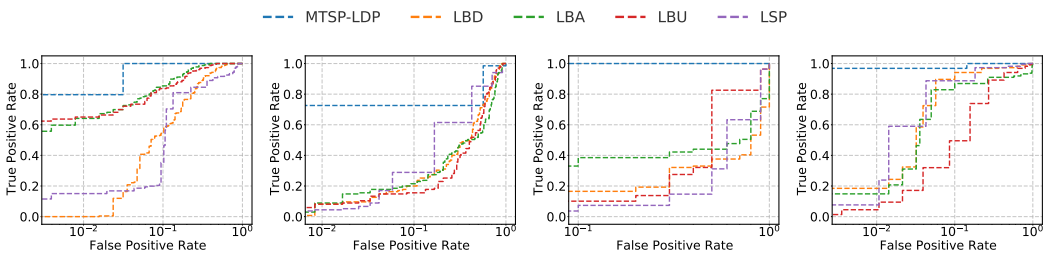
In contrast, MTSP-LDP's data-adaptive hierarchical structure and cross-timestamp processing enable more precise estimation for such queries, especially under tight privacy budgets.

5.3.3 Evaluation for Event Monitoring. For event monitoring, we define a specific task where the goal is to detect the change in the range query in the entire value domain Ω with threshold ϑ equal to the median of Ω in each dataset. The two functions in the range query are defined as follows.

$$\alpha(\Omega, \Delta) = Q_t^r(\Omega, \Delta) - Q_{t-w+1}^r(\Omega, \Delta)$$
$$\beta(x, \vartheta) = \mathbf{1}_{x>\vartheta}$$

By analyzing the changes across consecutive windows, we ensure a consistent comparison of each algorithm's ability to capture dynamic variations in the data streams. To better understand each method's performance, we vary the threshold ϑ, and obtain the ROC curve for each dataset, as illustrated in Fig. 8.

We observe that MTSP-LDP consistently outperforms the other methods, attaining near-perfect TPR across the full range of FPR. This demonstrates the robustness and adaptability of MTSP-LDP to different data distributions. At low FPR levels (FPR<0.1), MTSP-LDP achieves significantly higher



(a) Cosmetics (b) Taxi (c) Loan (d) Foursquare

Fig. 8. ROC for event monitoring with $\epsilon = 1$ and $w = 20$

TPR compared to other methods, highlighting its superior sensitivity under strict false positive constraints. In contrast, LBD and LBA exhibit limited performance, with slower TPR growth as FPR increases. While LSP shows relatively strong performance on the cosmetics dataset, approaching MTSP-LDP in some regions, it still falls short of MTSP-LDP's consistent accuracy across all datasets.

These results underscore the effectiveness of MTSP-LDP in addressing complex query tasks under varying privacy constraints and data characteristics. Its ability to achieve high accuracy across diverse datasets highlights its suitability for real-world applications requiring stringent privacy guarantees and robust performance.

6 Related Work

We summarize some related work in the literature , which can be classified into two categories, i.e., the CDP methods and the LDP methods.

6.1 CDP Methods

Event-level DP. Dwork et al. first introduced event-level DP for continuously releasing statistics like counts and histograms [17]. Hierarchical tree structures are widely used for range queries. In such trees, leaf nodes store the noisy counts for each timestamp, while internal nodes store noisy sums over the intervals they cover. For fixed-length binary streams, a binary tree structure was used to aggregate counts [18], and later extended to infinite streams by Chan et al. through consistency constraints [10]. To further reduce noise in sparse regions, Dwork et al. proposed adaptive partitioning based on thresholds [19]. However, this method is limited to post-partition release and lacks real-time applicability. Chen et al. [11] improved flexibility by modifying monitored events. Cao et al. [7] introduced group-based histogram publishing, adding Laplace noise to similar time slots. Although these techniques support numerical data, they require raw data access and thus cannot be directly applied under LDP. Bao et al. [6] addressed sliding-window predicate sum queries using decay models.

User-level DP. User-level DP frameworks focus on continuously publishing aggregate statistics. Fan et al. [24] proposed FAST, a real-time aggregation framework using adaptive sampling and filtering. It was extended to 2D monitoring via spatial partitioning with quadtrees [25]. Rastogi et al. [35] developed a method based on discrete Fourier transform (DFT), though it is suited only for offline analysis. The analysis in [14] reveals that on infinite streams, achieving user-level DP forces error to grow without bound as the maximum per-user prefix contribution increases over time. While they control privacy by truncating per-user contributions, this approach in turn introduces bias.

w-event-level DP. Kellaris et al. [28] introduced the w-event DP model to protect sequences of events over sliding windows and proposed BA/BD mechanisms. However, they rely on raw data to compute similarity, exposing them to inference attacks. Cao et al. [7] extended BD to support variable-length trajectories, using a greedy strategy to match current data with historical outputs. This resulted in uneven privacy budget distribution. Du et al. [15] extended BD/BA to the personalized w-event privacy setting, where different users have heterogeneous privacy requirements, and proposed PBD and PBA. Wang et al. [50] proposed E-RescueDP, which adaptively allocates privacy budgets using an RNN to support real-time release. Li et al. [30] proposed SPAS, which predicts future data variation to adaptively determine data sampling and privacy budget allocation under w-event DP.

6.2 LDP Methods

Most existing LDP works focus on single-value counting and frequency estimation of static data, with only a few studies focusing on stream data analysis. Memoization-based techniques [2, 13, 22] were proposed to offer longitudinal LDP guarantees. Joseph et al. [27] estimated stream means by having users compare their local averages against the last released value and vote on updates. Although satisfying event-level LDP, this method assumes Bernoulli input and temporal independence, limiting its generality. THRESH further assumes that the number of global updates is bounded by distribution changes, making it unsuitable for infinite streams. Wang et al. [46] proposed an event-level LDP framework for interval sum estimation using a hybrid mechanism with thresholding. However, it directly extends CDP-style hierarchies, offering limited protection for unbounded streams. Bao et al. [5] leveraged autocorrelation to reduce noise via an analytic Gaussian mechanism, but their method applies only to finite data and achieves approximate (ϵ, δ)-LDP under periodic budget renewal. Erlingsson et al. [21] introduced a shuffling model for correlated time series under user-level LDP, but their approach assumes integer-valued inputs with bounded updates, and the hierarchical design restricts scalability to infinite streams.

7 Conclusion

We propose MTSP-LDP, a w-event LDP framework to handle infinite data streams and support multiple streaming query tasks, including counting queries, range queries, and event monitoring. MTSP-LDP leverages a novel OBA algorithm that can dynamically allocate privacy budgets within a window. MTSP-LDP then constructs a private data-adaptive tree to support complex queries more accurately. Experiments on real-world datasets demonstrate that MTSP-LDP significantly outperforms state-of-the-art methods. Future work could focus on further enhancing MTSP-LDP, such as extending the framework to handle multidimensional data streams and optimizing its adaptability for even broader application scenarios.

Acknowledgments

This work was supported in part by National Key Research and Development Plan in China (2023YFC3306100) and National Natural Science Foundation of China (62272372).

References

[1] Fadi Al-Turjman, Hadi Zahmatkesh, and Ramiz Shahroze. 2022. An overview of security and privacy in smart cities' IoT communications. *Transactions on Emerging Telecommunications Technologies* 33, 3 (2022), 1–19.

[2] Héber Hwang Arcolezi, Jean-François Couchot, Bechara al Bouna, and Xiaokui Xiao. 2020. Longitudinal Collection and Analysis of Mobile Phone Data with Local Differential Privacy. In *Privacy and Identity Management: 15th IFIP WG 9.2, 9.6/11.7, 11.6/SIG 9.2.2 International Summer School, Maribor, Slovenia, September 21-23, 2020, Revised Selected Papers (IFIP Advances in Information and Communication Technology, Vol. 619)*, Michael Friedewald, Stefan Schiffner, and Stephan Krenn (Eds.). Springer, Cham, 40–57. doi:10.1007/978-3-030-72465-8_3

[3] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. 2002. Models and Issues in Data Stream Systems. In *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*, Lucian Popa, Serge Abiteboul, and Phokion G. Kolaitis (Eds.). ACM, New York, NY, USA, 1–16. doi:10.1145/543613.543615

[4] Shivnath Babu and Jennifer Widom. 2001. Continuous queries over data streams. *ACM SIGMOD Record* 30, 3 (2001), 109–120.

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

[6] Jean Bolot, Nadia Fawaz, S. Muthukrishnan, Aleksandar Nikolov, and Nina Taft. 2013. Private decayed predicate sums on streams. In *Joint 2013 EDBT/ICDT Conferences, ICDT '13 Proceedings, Genoa, Italy, March 18-22, 2013*, Wang-Chiew Tan, Giovanna Guerrini, Barbara Catania, and Anastasios Gounaris (Eds.). ACM, New York, NY, USA, 284–295. doi:10.1145/2448496.2448530

[7] Yang Cao and Masatoshi Yoshikawa. 2015. Differentially private real-time data release over infinite trajectory streams. In *16th IEEE International Conference on Mobile Data Management, MDM 2015, Pittsburgh, PA, USA, June 15-18, 2015 - Volume 2*, Christian S. Jensen, Xing Xie, Vladimir Zadorozhny, Sanjay Madria, Evaggelia Pitoura, Baihua Zheng, and Chi-Yin Chow (Eds.). IEEE Computer Society, Piscataway, NJ, USA, 68–73. doi:10.1109/MDM.2015.15

[8] Yang Cao, Masatoshi Yoshikawa, Yonghui Xiao, and Li Xiong. 2017. Quantifying Differential Privacy under Temporal Correlations. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*. IEEE Computer Society, Piscataway, NJ, USA, 821–832. doi:10.1109/ICDE.2017.132

[9] T.-H. Hubert Chan, Mingfei Li, Elaine Shi, and Wenchang Xu. 2012. Differentially Private Continual Monitoring of Heavy Hitters from Distributed Streams. In *Privacy Enhancing Technologies - 12th International Symposium, PETS 2012, Vigo, Spain, July 11-13, 2012. Proceedings (Lecture Notes in Computer Science, Vol. 7384)*, Simone Fischer-Hübner and Matthew K. Wright (Eds.). Springer, Cham, 140–159. doi:10.1007/978-3-642-31680-7_8

[10] T-H Hubert Chan, Elaine Shi, and Dawn Song. 2011. Private and continual release of statistics. *ACM Transactions on Information and System Security* 14, 3 (2011), 1–24.

[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, New York, NY, USA, 1375–1388. doi:10.1145/3133956.3134102

[12] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. 2018. Marginal Release Under Local Differential Privacy. 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, New York, NY, USA, 131–146. doi:10.1145/3183713.3196906

[13] 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.). Curran Associates, Inc., Red Hook, NY, USA, 3571–3580. https://proceedings.neurips.cc/paper/2017/hash/253614bbac999b38b5b60cae531c4969-Abstract.html

[14] 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, Piscataway, NJ, USA, 2190–2207. doi:10.1109/SP46215.2023.10179466

[15] Leilei Du, Peng Cheng, Lei Chen, Heng Tao Shen, Xuemin Lin, and Wei Xi. 2025. Infinite Stream Estimation under Personalized w-Event Privacy. *Proc. VLDB Endow.* 18, 6 (Aug. 2025), 1905–1918. doi:10.14778/3725688.3725715

[16] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. 2013. Local Privacy and Statistical Minimax Rates. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, Berkeley, CA, USA, October, 26-29, 2013*. IEEE Computer Society, Piscataway, NJ, USA, 429–438. doi:10.1109/FOCS.2013.53

[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, Cham, 265–284. doi:10.1007/11681878_14

[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, New York, NY, USA, 715–724. doi:10.1145/1806689.1806787

[19] Cynthia Dwork, Moni Naor, Omer Reingold, and Guy N. Rothblum. 2015. Pure Differential Privacy for Rectangle Queries via Private Partitions. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 9453)*, Tetsu Iwata and Jung Hee Cheon (Eds.). Springer,

Cham, 735–751. doi:10.1007/978-3-662-48800-3_30

[20] Cynthia Dwork and Aaron Roth. 2013. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3-4 (2013), 211–407.

[21] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. 2019. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, Timothy M. Chan (Ed.). SIAM, Philadelphia, PA, USA, 2468–2479. doi:10.1137/1.9781611975482.151

[22] 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, New York, NY, USA, 1054–1067. doi:10.1145/2660267.2660348

[23] 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, New York, NY, USA, 2169–2173. doi:10.1145/2396761.2398595

[24] Liyue Fan and Li Xiong. 2013. An adaptive approach to real-time aggregate monitoring with differential privacy. *IEEE Transactions on Knowledge and Data Engineering* 26, 9 (2013), 2094–2106.

[25] Liyue Fan, Li Xiong, and Vaidy S. Sunderam. 2013. Differentially Private Multi-dimensional Time Series Release for Traffic Monitoring. In *Data and Applications Security and Privacy XXVII - 27th Annual IFIP WG 11.3 Conference, DBSec 2013, Newark, NJ, USA, July 15-17, 2013. Proceedings (Lecture Notes in Computer Science, Vol. 7964)*, Lingyu Wang and Basit Shafiq (Eds.). Springer, Cham, 33–48. doi:10.1007/978-3-642-39256-6_3

[26] James Honaker. 2015. Efficient Use of Differentially Private Binary Trees. In *Theory and Practice of Differential Privacy*. TPDP, London, UK, 1–1.

[27] Matthew Joseph, Aaron Roth, Jonathan R. Ullman, and Bo Waggoner. 2018. Local Differential Privacy for Evolving Data. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). Curran Associates, Inc., Red Hook, NY, USA, 2381–2390. https://proceedings.neurips.cc/paper/2018/hash/a01610228fe998f515a72dd730294d87-Abstract.html

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

[29] Tejas Kulkarni. 2019. Answering Range Queries Under Local Differential Privacy. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, Peter Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska (Eds.). ACM, New York, NY, USA, 1832–1834. doi:10.1145/3299869.3300102

[30] Xiaochen Li, Tianyu Li, Yitian Cheng, Chen Gong, Kui Ren, Zhan Qin, and Tianhao Wang. 2025. SPAS: Continuous Release of Data Streams under w-Event Differential Privacy. *Proc. ACM Manag. Data* 3, 1 (2025), 78a:1–78a:27. doi:10.1145/3714420

[31] Parushi Malhotra, Yashwant Singh, Pooja Anand, Deep Kumar Bangotra, Pradeep Kumar Singh, and Wei-Chiang Hong. 2021. Internet of things: Evolution, concerns and security challenges. *Sensors* 21, 5 (2021), 1–33.

[32] Takao Murakami and Yusuke Kawamoto. 2019. Utility-Optimized Local Differential Privacy Mechanisms for Distribution Estimation. In *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, Nadia Heninger and Patrick Traynor (Eds.). USENIX Association, Berkeley, CA, USA, 1877–1894. https://www.usenix.org/conference/usenixsecurity19/presentation/murakami

[33] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2016. Heavy Hitter Estimation over Set-Valued Data with Local Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi (Eds.). ACM, New York, NY, USA, 192–203. doi:10.1145/2976749.2978409

[34] Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2017. Generating Synthetic Decentralized Social Graphs with Local Differential Privacy. 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, New York, NY, USA, 425–438. doi:10.1145/3133956.3134086

[35] Vibhor Rastogi and Suman Nath. 2010. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, Ahmed K. Elmagarmid and Divyakant Agrawal (Eds.). ACM, New York, NY, USA, 735–746. doi:10.1145/1807167.1807247

[36] Partha Pratim Ray, Dinesh Dash, and Neeraj Kumar. 2020. Sensors for internet of medical things: State-of-the-art, security and privacy issues, challenges and future directions. *Computer Communications* 160 (2020), 111–131.

[37] Xuebin Ren, Liang Shi, Weiren Yu, Shusen Yang, Cong Zhao, and Zongben Xu. 2022. LDP-IDS: Local Differential Privacy for Infinite Data Streams. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, New York, NY, USA, 1064–1077. doi:10.1145/3514221.3526190

[38] Xuebin Ren, Chia-Mu Yu, Weiren Yu, Shusen Yang, Xinyu Yang, Julie A McCann, and Philip S Yu. 2018. LoPub: high-dimensional crowdsourced data publication with local differential privacy. *IEEE Transactions on Information Forensics and Security* 13, 9 (2018), 2151–2166.

[39] Xuebin Ren, Chia-Mu Yu, Wei Yu, Xinyu Yang, Jun Zhao, and Shusen Yang. 2020. DPCrowd: Privacy-preserving and communication-efficient decentralized statistical estimation for real-time crowdsourced data. *IEEE Internet of Things Journal* 8, 4 (2020), 2775–2791.

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

[41] Abida Sharif, Jianping Li, Mudassir Khalil, Rajesh Kumar, Muhammad Irfan Sharif, and Atiqa Sharif. 2017. Internet of things—smart traffic management system for smart cities using big data analytics. In *2017 14th international computer conference on wavelet active media technology and information processing (ICCWAMTIP)*. IEEE, IEEE, New York, NY, USA, 281–284.

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

[43] Ning Wang, Xiaokui Xiao, Yin Yang, Ta Duy Hoang, Hyejin Shin, Junbum Shin, and Ge Yu. 2018. PrivTrie: Effective Frequent Term Discovery under Local Differential Privacy. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*. IEEE Computer Society, New York, NY, USA, 821–832. doi:10.1109/ICDE.2018.00079

[44] 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, Piscataway, NJ, USA, 1–9. doi:10.1109/INFOCOM.2016.7524458

[45] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. 2017. Locally Differentially Private Protocols for Frequency Estimation. In *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*, Engin Kirda and Thomas Ristenpart (Eds.). USENIX Association, Berkeley, CA, USA, 729–745. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-tianhao

[46] 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, New York, NY, USA, 1237–1253. doi:10.1145/3460120.3484750

[47] Tianhao Wang, Bolin Ding, Jingren Zhou, Cheng Hong, Zhicong Huang, Ninghui Li, and Somesh Jha. 2019. Answering Multi-Dimensional Analytical Queries under Local Differential Privacy. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, Peter Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska (Eds.). ACM, New York, NY, USA, 159–176. doi:10.1145/3299869.3319891

[48] Tianhao Wang, Ninghui Li, and Somesh Jha. 2019. Locally differentially private heavy hitter identification. *IEEE Transactions on Dependable and Secure Computing* 18, 2 (2019), 982–993.

[49] Zhibo Wang, Wenxin Liu, Xiaoyi Pang, Ju Ren, Zhe Liu, and Yongle Chen. 2020. Towards Pattern-aware Privacy-preserving Real-time Data Collection. In *39th IEEE Conference on Computer Communications, INFOCOM 2020, Toronto, ON, Canada, July 6-9, 2020*. IEEE, Piscataway, NJ, USA, 109–118. doi:10.1109/INFOCOM41043.2020.9155290

[50] Zhibo Wang, Xiaoyi Pang, Yahong Chen, Huajie Shao, Qian Wang, Libing Wu, Honglong Chen, and Hairong Qi. 2019. Privacy-Preserving Crowd-Sourced Statistical Data Publishing with An Untrusted Server. *IEEE Transactions on Mobile Computing* 18, 6 (2019), 1356–1367.

[51] Qingqing Ye, Haibo Hu, Xiaofeng Meng, and Huadi Zheng. 2019. PrivKV: Key-Value Data Collection with Local Differential Privacy. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, Piscataway, NJ, USA, 317–331. doi:10.1109/SP.2019.00018

[52] Zhikun Zhang, Tianhao Wang, Ninghui Li, Shibo He, and Jiming Chen. 2018. CALM: Consistent Adaptive Local Marginal for Marginal Release under Local Differential Privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang (Eds.). ACM, New York, NY, USA, 212–229. doi:10.1145/3243734.3243742

Received July 2025; revised October 2025; accepted November 2025