
Differential Privacy on Fully Dynamic Streams

Yuan Qiu

School of Cyber Science and Engineering
Southeast University
Nanjing, China 211189
yuanqiu@seu.edu.cn

Ke Yi

Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong SAR, China 999077
yike@cse.ust.hk

Abstract

A fundamental problem in differential privacy is to release privatized answers to a class of linear queries with small error. This problem has been well studied in the static case. In this paper, we consider the fully dynamic setting where items may be inserted into or deleted from the dataset over time, and we need to continually release query answers at every time instance. We present efficient black-box constructions of such dynamic differentially private mechanisms from static ones with only a polylogarithmic degradation in the utility.

1 Introduction

In the data streaming model, a stream consists of a pair of possibly infinitely long vectors $(\vec{s}, \vec{x})$. It defines a dynamically changing dataset D_t , which is a multiset of elements from a universe $\mathcal{X}$. Initially, $D_0 = \emptyset$. At time $t \in \mathbb{N}^+$, (s_t, x_t) arrives, where $s_t \in \{+, -, \perp\}$ indicates the operation and $x_t \in \mathcal{X}$ is the relevant element. Then D_t is defined inductively as follows:

- If $s_t = +$, then $D_t := D_{t-1} \uplus \{x_t\}$, i.e., one copy of x_t is inserted into the dataset.
- If $s_t = -$, then $D_t := D_{t-1} - \{x_t\}$, i.e., one copy of x_t is deleted from the dataset (it is required that at least one copy of x_t exists in D_{t-1}).
- If $s_t = \perp$, then $D_t := D_{t-1}$.

Two variants of the streaming model have been considered in the literature [25, 5]: The model above is referred to as the fully dynamic setting (a.k.a. the *turnstile model*), while the case where s_t can only be $+$ or $\perp$ is called the insertion-only setting (a.k.a. the *cash register model*).

A fundamental problem in differential privacy (DP) is how to answer a class of *linear queries* on a given dataset D privately and accurately. In particular, the *private multiplicative weights mechanism* (PMW) [15, 17] can answer a set of arbitrary linear queries with maximum error¹ $\tilde{O}(\sqrt{|D|})$. Better error bounds can be achieved if the class of queries exhibit good structures. For instance, d -dimensional half-space queries can be answered with error $\tilde{O}(|D|^{\frac{1}{2} - \frac{1}{2d}})$ [26]; please see Appendix A for a review of results on various query classes. Note that, since there can be many queries in the class, these algorithms do not return the query answers explicitly; instead, a privatized data structure (e.g., a histogram in the case of PMW) is often returned, and a data analyst can subsequently extract the answer of any query from the data structure.

Combining the query-answering problem and the data streaming model naturally gives rise to the problem of *differential privacy under continual observation*, first studied in the foundational work of Dwork et al. [9]. Here, the goal is to continually release privatized query answers under the following requirements:

¹The $\tilde{O}$ notation suppresses dependencies on the privacy parameter ϵ and polylogarithmic terms.

- **Online:** For any t , we must release a data structure $\mathcal{M}^{(t)}(D_t)$ before (s_{t+1}, x_{t+1}) arrives.
- **Private:** All released data structures $(\mathcal{M}^{(1)}(D_1), \mathcal{M}^{(2)}(D_2), \dots)$ jointly satisfy DP.
- **Accurate:** For every t , all queries on D_t can be answered from $(\mathcal{M}^{(1)}(D_1), \dots, \mathcal{M}^{(t)}(D_t))$ with small error, ideally matching the error in the static case, i.e., as if all queries were answered on D_t one-shot.
- **Efficient:** The mechanism should run efficiently.

The pioneer work [9, 4] has only studied the basic counting problem, i.e., the query just asks for $|D_t|$, under the insertion-only streaming model. However, we observe that their algorithms are actually black-boxed, and can be instantiated with any static mechanism for some other query class, as long as the queries are union-preserving, with a polylogarithmic-factor increase in the error bound. Formally, a query f is called union-preserving if $f(D^{(1)} \uplus D^{(2)}) = f(D^{(1)}) + f(D^{(2)})$ for any datasets $D^{(1)}, D^{(2)}$, where $\uplus$ denotes the multiset union. For example, by plugging in PMW, any class of linear queries on D_t can be answered with error $\tilde{O}(\sqrt{|D_t|})$ for every t . Perhaps not realizing this, [6] presented a white-box version of PMW for the insertion-only streaming model, but the error is $\tilde{O}(|D_t|^{3/4})$. They also showed a black-box solution, but the error is $\tilde{O}(|D_t|^{5/6})$ when instantiated with PMW.

A standard approach to dealing with the fully-dynamic case is to divide the stream into two insertion-only streams: one only containing insertions and one only containing deletions but treating these deletions as insertions. Let $D_t^+ := \uplus_{i \leq t: s_i = +} \{x_i\}$ be all items inserted up to time t , and $D_t^- := \uplus_{i \leq t: s_i = -} \{x_i\}$ all items deleted up to time t , then $D_t = D_t^+ - D_t^-$. For any union-preserving query f , we have $f(D_t) = f(D_t^+) - f(D_t^-)$, so we can run two separate instances of the insertion-only mechanism, and use their difference to answer f on D_t . However, the worst-case error of this solution is very bad. Suppose we instantiate the insertion-only algorithm of [9, 4] with PMW. Then $f(D_t^+)$ and $f(D_t^-)$ can be answered with error $\tilde{O}(\sqrt{|D_t^+|})$ and $\tilde{O}(\sqrt{|D_t^-|})$, respectively, which means that the error for $f(D_t)$ is $\tilde{O}(\sqrt{|D_t^+| + |D_t^-|})$. This can be arbitrarily worse than the optimal error of $\tilde{O}(\sqrt{|D_t|})$ in the static case: D_t^+ and D_t^- may both be very large but $D_t = \emptyset$.

Our Contributions. In this paper, we present a black-box algorithm for fully-dynamic streams that can answer any class of union-preserving queries with error matching that in the static case, up to polylogarithmic factors. For instance, when instantiated with PMW, our algorithm achieves an error of $\tilde{O}(\sqrt{|D_t|})$ for every t . Furthermore, its total running time up to time t is also just a polylogarithmic factor higher than that of the static mechanism run on all the stream elements up to time t .

2 Preliminaries

2.1 Differential Privacy

Let $\mathcal{X}$ be the universe of items. A static dataset is a multiset of items $D \in \mathbb{N}^{\mathcal{X}}$. Two static datasets $D, D' \in \mathbb{N}^{\mathcal{X}}$ are *neighbors*, denoted $D \sim D'$, if there exists an item $x \in \mathcal{X}$, such that $D = D' \uplus \{x\}$ or vice versa. *Differential privacy (DP)* [8] is defined as follows.

Definition 1 (Differential Privacy [8]). *A randomized mechanism $\mathcal{M} : \mathbb{N}^{\mathcal{X}} \rightarrow \mathcal{Y}$ satisfies (ϵ, δ) -DP if for any neighboring datasets $D \sim D'$ and any subset of outputs $Y \subseteq \mathcal{Y}$,*

$$\Pr[\mathcal{M}(D) \in Y] \leq e^\epsilon \cdot \Pr[\mathcal{M}(D') \in Y] + \delta. \quad (1)$$

In the streaming setting, two streams are neighbors if one has one more update, which can be either an insertion or a deletion, than the other [9]. Formally, two streams $(\vec{s}, \vec{x})$ and $(\vec{s}', \vec{x}')$ are neighbors, if $\exists i \in \mathbb{N}^+$ such that $(s_j, x_j) = (s'_j, x'_j)$ for any $j \neq i$, and either s_i or s'_i is $\perp$. Plugging this neighboring relationship into Definition 1 yields the DP definition for the streaming model. More precisely, $\mathcal{M}(D)$ and $\mathcal{M}(D')$ in (1) are replaced by $(\mathcal{M}^{(1)}(D_1), \mathcal{M}^{(2)}(D_2), \dots)$ and $(\mathcal{M}^{(1)}(D'_1), \mathcal{M}^{(2)}(D'_2), \dots)$ respectively, where D_i and D'_i are the datasets induced by any two neighboring streams at time i .

An important property used in designing DP mechanisms is composition, which comes in two settings:

Theorem 2 (Sequential Composition [8]). *Let $\mathcal{M}_i : \mathbb{N}^{\mathcal{U}} \rightarrow \mathcal{Y}_i$ each be an $(\varepsilon_i, \delta_i)$ -DP mechanism. Then the composed mechanism $\mathcal{M}(D) = (\mathcal{M}_1(D), \dots, \mathcal{M}_k(D))$ is $(\sum_{i=1}^k \varepsilon_i, \sum_{i=1}^k \delta_i)$ -DP.*

Note that there are many “advanced” versions of sequential composition [10, 21, 2] with better dependencies on k . Nevertheless, as k is logarithmic in all of our algorithms, those advanced versions do not offer better bounds for the problem studied in this paper.

Theorem 3 (Parallel Composition [24]). *Let $\mathcal{U} = \mathcal{U}_1 \cup \dots \cup \mathcal{U}_k$ be a partitioning of the universe² $\mathcal{U}$, and $\mathcal{M}_i : \mathbb{N}^{\mathcal{U}_i} \rightarrow \mathcal{Y}_i$ each be an $(\varepsilon_i, \delta_i)$ -DP mechanism. Then the composed mechanism $\mathcal{M}(D) = (\mathcal{M}_1(D \cap \mathcal{U}_1), \dots, \mathcal{M}_k(D \cap \mathcal{U}_k))$ is $(\max_{i=1}^k \varepsilon_i, \max_{i=1}^k \delta_i)$ -DP.*

2.2 Linear Queries

A *linear query* is specified by a function $f : \mathcal{X} \rightarrow [0, 1]$. The result of evaluating f on D is defined as $f(D) := \sum_{x \in D} f(x)$. A fundamental problem in differential privacy is the following: Given a set of linear queries $\mathcal{F} = \{f_1, \dots, f_{|\mathcal{F}|}\}$, design a DP mechanism $\mathcal{M}$ that, on any given D , outputs a data structure $\mathcal{M}(D)$, from which an approximate $\hat{f}_{\mathcal{M}}(D)$ can be extracted for any $f \in \mathcal{F}$. We say that $\mathcal{M}$ has error α with probability $1 - \beta$, if

$$\Pr \left[\max_{f \in \mathcal{F}} \left| \hat{f}_{\mathcal{M}}(D) - f(D) \right| > \alpha \right] \leq \beta$$

for any D , where the probability is taken over the internal randomness of $\mathcal{M}$. Clearly, α is a function of the privacy parameters (ε, δ) and the failure probability β . For most mechanisms, it also depends on the data size $|D|$, number of queries $|\mathcal{F}|$, and domain size $|\mathcal{X}|$. To simplify notation, we often omit some of these parameters from the full list $\alpha(\varepsilon, \delta, \beta, |D|, |\mathcal{F}|, |\mathcal{X}|)$ if they are clear from the context. Similarly, we denote the running time of the mechanism by $\text{time}(|D|, \cdot)$, which depends on the data size $|D|$ and possibly other parameters.

There is extensive work [30, 15, 17, 29, 14, 1, 22, 23, 26] on the best achievable α for various families of linear queries. This paper takes a black-box approach, i.e., we present dynamic algorithms that can work with any $\mathcal{M}$ that has been designed for queries $\mathcal{F}$ on a static dataset D . The error for the dynamic algorithm will be stated in terms of the α function of the mechanism $\mathcal{M}$ that is plugged into the black box. Nevertheless, we often derive the explicit bounds for the following two most interesting and extreme cases:

Basic counting. If $\mathcal{F}$ consists of a single query $f(\cdot) \equiv 1$, which simply returns $f(D) = |D|$, then the “data structure” $\mathcal{M}(D)$ consists of just one number, which is a noise-masked $f(D)$. The most popular choice of the noise is a random variable drawn from the Laplace distribution $\text{Lap}(\frac{1}{\varepsilon})$. Its error function α is given by $\alpha_{\text{Lap}}(\varepsilon, \beta) = O\left(\frac{1}{\varepsilon} \log \frac{1}{\beta}\right)$. Alternatively, one can add a Gaussian noise, which is (ε, δ) -DP for $\delta > 0$ and yields $\alpha_{\text{Gauss}}(\varepsilon, \delta, \beta) = O\left(\frac{1}{\varepsilon} \sqrt{\log \frac{1}{\delta} \log \frac{1}{\beta}}\right)$. The two error bounds are generally incomparable, but the former is usually better since $\delta \leq \beta$ in common parameter regimes. Both the Laplace and the Gaussian mechanism have $\text{time}(|D|) = O(|D|)$.

Arbitrary queries. If $\mathcal{F}$ is a set of arbitrary linear queries, then the *private multiplicative weights* (PMW) [15, 17] mechanism achieves $\alpha_{\text{PMW}} = \tilde{O}(\sqrt{|D|})$ for $\delta > 0$ and $\tilde{O}(|D|^{2/3})$ for $\delta = 0$. These error bounds are optimal for $|\mathcal{F}|$ sufficiently large. The running time of PMW is $\tilde{O}(|D| + |\mathcal{X}| \cdot |\mathcal{F}| \cdot |D|^2 / \alpha^2)$.

There are many possibilities between the two extreme cases, and the achievable error bound α intricately depends on the discrepancy of the query set $\mathcal{F}$ [16, 26]. We include a brief review in Appendix A, which is not necessary for the understanding of this paper. We make a reasonable assumption that α does not depend on any of its parameters exponentially, which allows us to ignore the constant coefficients in the parameters, e.g., $\alpha(O(\varepsilon))$ is asymptotically the same as $\alpha(\varepsilon)$. This assumption holds for most existing mechanisms for linear queries.

²Note that the “universe” $\mathcal{U}$ above is not necessarily the same as the universe $\mathcal{X}$ of stream items. In fact, in our algorithms, we often take $\mathcal{U}$ as the time domain, and parallel composition then implies that running an (ε, δ) -DP mechanism in each disjoint time interval satisfies (ε, δ) -DP over the entire stream.

2.3 Differential Privacy for Insertion-only Streams

The insertion-only algorithms, as well as our fully dynamic algorithm, work by decomposing $D = D^{(1)} \uplus \dots \uplus D^{(k)}$ for some k , so $f(D) = f(D^{(1)}) + \dots + f(D^{(k)})$ by the union-preserving property. Thus, if the error of each $f(D^{(i)})$ is $\alpha(\cdot, \beta)$, the error of $f(D)$ is at most $k \cdot \alpha(\cdot, \beta/k)$ by a union bound. Further, if the estimates of $f(D^{(i)})$ are unbiased with good concentration properties, this bound can be tightened. For example, for the basic counting query $f(D) = |D|$, if we use the Laplace mechanism to estimate each $f(D^{(i)})$, then the error for $f(D)$ is $O\left(\frac{1}{\varepsilon} \left(\sqrt{k \log \frac{1}{\beta}} + \log \frac{1}{\beta}\right)\right)$,

which is better than $k \cdot \alpha_{\text{Lap}}(\cdot, \beta/k) = O\left(\frac{k}{\varepsilon} \log \frac{k}{\beta}\right)$. We derive this result, as well as the error bounds of some other mechanisms under such a disjoint union, in Appendix B. In the main text, for generality we will use $\alpha^{(k)}$ to denote this error bound. Note that $\alpha^{(k)}$ is also a function of $|D|, \varepsilon, \delta, \beta, |\mathcal{F}|, |\mathcal{X}|$, but we may omit these parameters when the context is clear. Using this notation, the result of existing insertion-only algorithms can be restated as follows.

Lemma 4 ([4]). *For a set of union-preserving queries $\mathcal{F}$, let $\mathcal{M} : \mathbb{N}^{\mathcal{X}} \rightarrow \mathcal{Y}$ be an (ε, δ) -DP static mechanism whose error is $\alpha(\varepsilon, \delta, \beta, |D|, |\mathcal{F}|, |\mathcal{X}|)$ on any D , with running time $\text{time}(|D|)$. Then there is an (ε, δ) -DP mechanism $\mathcal{M}_{\mathcal{F}, \text{ins}}$ for insertion-only streams, so that at any time t , it answers queries $\mathcal{F}$ on D_t with error*

$$\gamma(t; \varepsilon, \delta, \beta, |D_t|, |\mathcal{F}|, |\mathcal{X}|) = \alpha^{(\log t)}\left(\frac{\varepsilon}{\log t}, \frac{\delta}{\log t}, \beta, |D_t|, |\mathcal{F}|, |\mathcal{X}|\right).$$

The total running time³ of $\mathcal{M}_{\mathcal{F}, \text{ins}}$ up to time t is $O(\log t) \cdot \text{time}(|D_t|)$.

Our algorithm will also use a special case of Lemma 4 when $\mathcal{F}$ is the basic counting query. Plugging in our bound on $\alpha_{\text{Lap}}^{(k)}$ in Appendix B leads to the following explicit error bound, which is slightly better than that in [4].

Corollary 1. *There is an ε -DP mechanism $\mathcal{M}_{\text{cnt}, \text{ins}}$ that at any time t , answers the counting query $f(D_t) = |D_t|$ on an insertion-only stream with error*

$$\gamma_{\text{cnt}}(t; \varepsilon, \beta) = \alpha_{\text{Lap}}^{(\log t)}\left(\frac{\varepsilon}{\log t}, \beta\right) = O\left(\frac{\log^{1.5} t}{\varepsilon} \sqrt{\log \frac{1}{\beta}} + \frac{\log t}{\varepsilon} \log \frac{1}{\beta}\right).$$

3 Differential Privacy on Fully Dynamic Streams

In this section, we describe our algorithm for answering a set of union-preserving queries on a possibly infinite fully dynamic stream under differential privacy. We use $n_t := |D_t|$ to denote the size of the dataset at time t , and N_t to denote the total number of updates up to time t . Note that for an insertion-only stream we have $n_t = N_t$, but on a fully dynamic stream n_t may be much smaller than N_t . Our goal is to achieve an error bound that depends on n_t .

We will treat a fully dynamic stream as a set of labeled time intervals. An interval $[i, j)$ labeled with item $x \in \mathcal{X}$ represents that a copy of x is inserted at time i and deleted at time j . It is possible that $j = \infty$, if the item is never deleted. Note that this interval representation of a stream is not unique, e.g., when many copies of the same item are inserted and then deleted. Any representation can be used; in fact, our algorithm does not use this interval representation, only the analysis does.

Using the interval representation, D_t consists of all items whose intervals are stabbed by t , i.e., all intervals $[i, j) \ni t$. We will make use of the *interval tree* [7] to organize all the intervals. In an interval tree, each D_t is decomposed into a logarithmic number of subsets, each of which consists of one-sided intervals, which will allow us to use the insertion-only mechanism. However, there are two technical difficulties in implementing such a plan. First, the intervals are given in an online fashion, i.e., at time t , we only see the endpoints of the intervals prior to t . When we see the left endpoint of an interval, we do not know where in the interval tree to put this interval, yet, we need to immediately release privatized information about this interval. Second, the interval tree on an infinite stream is also infinitely large, so we have to build it incrementally, while allocating the privacy budget appropriately.

³For running time, it is often assumed that no-ops in the stream can be skipped at no cost.

We describe how to overcome the difficulties above in Section 3.1. In Section 3.2, we introduce a DP mechanism running at each node of the new tree structure to support querying at any time with respect to intervals stored in the tree. The output of the whole mechanism is obtained by combining the individual mechanisms at the tree nodes, which is summarized in Section 3.3.

3.1 Online Interval Tree

We first build a binary tree $\mathcal{T}$ over the timestamps, where each timestamp corresponds to a tree node. Figure 1 shows an (offline) interval tree built on the first 8 timestamps. It is clear that the tree on timestamps $\{1, \dots, t\}$ has t nodes and $O(\log t)$ height. In the online setting, $\mathcal{T}$ will grow from left to right. We order the nodes using an in-order traversal of $\mathcal{T}$: $v_1, v_2, \dots$, and we will release the information about v_i right after time i .

Ignoring differential privacy for now, let us first focus on how to answer a stabbing query using an interval tree, i.e., report all items (intervals) in the dataset D_q at query time q . In a standard interval tree, an interval is stored at the *highest* node v_t such that its timestamp t stabs the interval. Denote by $D(v)$ the set of labeled intervals stored at v . For example, $D(v_4) = \{a, b, c\}$ in Figure 1. Given a query at time q , we follow the root-to-node path to v_q in $\mathcal{T}$. For each *left* node v_l on the path where $l \leq q$, we find all intervals $[i, j]$ in $D(v_l)$ such that $j > q$; for each *right* node v_r on the path where $r > q$, we find all intervals $[i, j]$ in $D(v_r)$ such that $i \leq q$. Standard analysis on the interval tree shows that these subsets form a disjoint union of all intervals stabbed by q .

Example 1 (Interval Tree Query). *Given the interval tree in Figure 1, assume a query is issued at time $q = 6$. We follow the path $(v_6, v_4, v_8, \dots)$. Along the path, v_4 is on the left, where $a, c \in D(v_4)$ are deleted after $q = 6$; v_8 is on the right, where $d \in D(v_8)$ begins at (or before) $q = 6$. Thus we report $D_6 = \{a, c, d\}$, which are exactly the elements present in the dataset at the query time.*

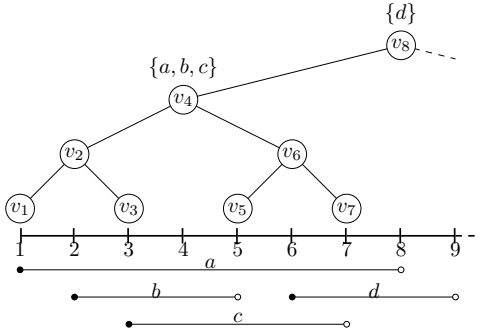


Figure 1: A standard interval tree.

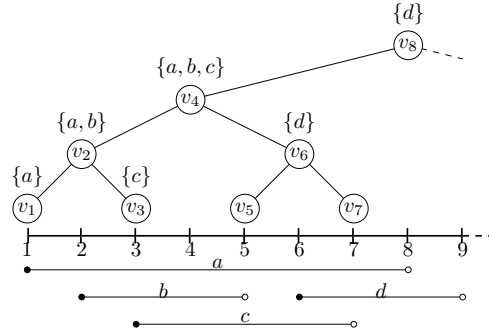


Figure 2: An online interval tree.

In an online setting, however, we do not know which node is the highest to put an interval in, since we do not know the deletion time when an item is inserted. Consider timestamp 3 in Figure 1, where item c is inserted into $D_2 = \{a, b\}$. If c were to be deleted at timestamp 4, then v_3 becomes the highest node that stabs it, which by definition should store c . However, if c is deleted at some timestamp among $\{5, 6, 7\}$, then v_4 is the highest node. Other possible candidates are $v_8, v_{16}, \dots$. One idea is to put a copy of the interval into every node where the interval *might* be placed into. But there are infinitely many such nodes, therefore we do so lazily. More formally, we design a novel *online interval tree* that is capable of handling an infinite stream.

Definition 5 (Online Interval Tree). *In an online interval tree $\mathcal{T}$, a tree node v_t stores an interval $[i, j]$ if and only if both A) $i \leq t < j$; and B) v_i is in the subtree rooted at v_t .*

Compared with the standard offline interval tree, an online interval tree may store an interval $[i, j]$ at multiple nodes. Nevertheless, condition B) implies that all these nodes lie on the root-to-node path to v_i , so at each level, there is at most one node that stores $[i, j]$, which is the key property we will need. We use Figure 2 to illustrate. Interval a is stored at v_1, v_2, v_4 , which are ancestors of v_1 . It is not stored in v_8 since condition A) is violated: intuitively, by timestamp 8, a is already deleted, so there is no need to store a at v_8 . On the other hand, a needs to be stored in all v_1, v_2, v_4 (in the standard interval tree, it is only stored at v_4), because by timestamp 1 or 2, we still do not know the deletion time of a . Similarly, interval d is stored in both v_6 and v_8 .

3.1.1 Building the Online Interval Tree

This online interval tree can be incrementally constructed easily. After observing the update at timestamp t , we first compute the dataset D_t at time t . These are exactly the elements that satisfy condition A). Then we construct the dataset $D(v_t)$ for the node v_t out of elements in D_t , keeping only those that also satisfy condition B). These will be the intervals whose insertion-time node $v_i = v_t$ or lies in the left-subtree of v_t . For any node on the left-most path of $\mathcal{T}$ (where t is a power of 2), $D(v_t) = D_t$ simply contains all elements in the current dataset. Otherwise, $D(v_t) \subseteq D_t$ will only contain items inserted after its closest left-ancestor.

Example 2 (Online Interval Tree Construction). In Figure 2, we construct $D(v_4) = D_4 = \{a, b, c\}$ at $t = 4$. To construct $D(v_6)$, we first compute $D_6 = \{a, c, d\}$. But we only consider items whose insertion-time is in the left-subtree (namely $\{v_5, v_6\}$), therefore $D(v_6) = \{d\}$ will only include d from D_6 . The intuition is that a and c have already been covered by v_4 .

Note that when $D(v_t)$ is first constructed, we do not have the deletion times of the items in $D(v_t)$, which will be added when these items are actually deleted later. For example, in Figure 2, $D(v_2) = \{a, b\}$ is constructed at timestamp 2 but neither item is associated with a deletion time. After timestamp 5, we add the deletion time of b , augmenting $D(v_2)$ to $\{a, (b, 5)\}$; after timestamp 8, it becomes $\{(a, 8), (b, 5)\}$. Note that there is no need to associate the left endpoints (i.e., insertion times) to the items as in the standard interval tree, and we will see why below.

3.1.2 Querying the Online Interval Tree

Now we show how to answer a stabbing query using the online interval tree. Since the online interval tree includes multiple copies of an item, the standard interval tree query algorithm will not work, as it may report duplicates. For the stabbing problem itself, duplicates are not an issue as they can be easily removed if they have been reported already. However, for answering linear queries, we actually need to cover all stabbed intervals by a disjoint union of subsets. To achieve it, we modify the stabbing query process as follows. Given a query at time q . We follow the root-to-node path to v_q in $\mathcal{T}$, and only consider *left* nodes v_l on the path where $l \leq q$. For each v_l , we report all intervals $[i, j]$ in $D(v_l)$ where $j > q$.

Example 3 (Online Interval Tree Query). Again consider a query at $q = 6$ in Figure 2, on the root-to-node path, nodes v_4 and v_6 satisfy $l \leq q$. We visit them and report $\{a, c\}$ and $\{d\}$ respectively, which jointly form the dataset D_6 .

Unlike in the standard interval tree, we do not query those *right* ancestors (e.g. v_8). It turns out that the items stored in the right ancestors are exactly compensated by the extra copies of items stored in the left ancestors of v_q . The following lemma formalizes this guarantee.

Lemma 6. *The query procedure described above reports each stabbed interval exactly once.*

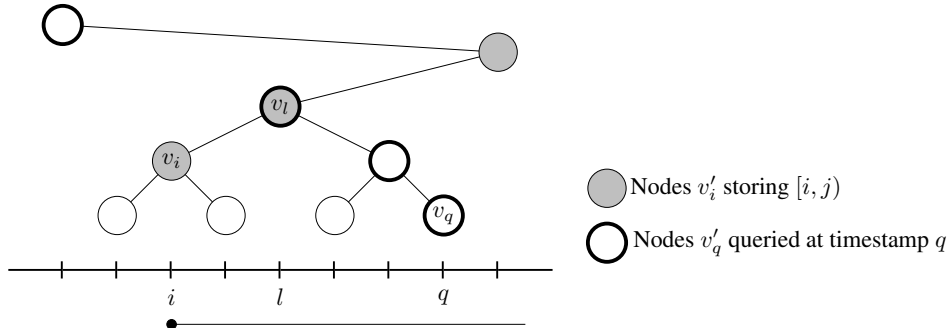


Figure 3: Querying a stabbed interval.

Proof. Given a query at time q , consider any item inserted at time i and deleted at time j , represented by interval $[i, j]$. We first prove that this item will not be reported if it is not in the dataset D_q , i.e. interval $[i, j]$ is not stabbed by q . This happens when: (1) the item has been deleted at query time ($j \leq q$). As we only report an item whose deletion time $j > q$, the interval is filtered out; (2) the item

has not arrived by query time ($i > q$). As we only visit *left* nodes v_l where $l \leq q$, it follows that $l \leq q < i$. By definition, v_l can store an item only if $l \geq i$, so this item is not stored by v_l .

The final case is when $i \leq q < j$, and the interval is supposed to be reported by exactly one node. This is shown in Figure 3. For the trivial case that $q = i$, the newly constructed node v_i is the only node reporting this interval. Otherwise, consider the minimum subtree containing both v_i and v_q , and assume it is rooted at v_l . We must have v_i in its left subtree and v_q in its right subtree by the minimum property, with the only exception that one of them can be v_l itself, i.e., $i \leq l \leq q$. We can argue that v_l is the only node that reports the interval: any node $v'_i \neq v_l$ that stores $[i, j]$ is either in the left subtree of v_l , or an ancestor of v_l that is on the right side of v_l ; any node $v'_q \neq v_l$ queried at q is either in the right subtree of v_l , or an ancestor of v_l that is on the left side of v_l . Thus the only node that can possibly report this interval is v_l . Since v_l is queried and $j > q$, this stabbing interval is reported exactly once by v_l . $\square$

3.2 Deletion-only Mechanism at Each Node

We have shown that the online interval tree can be incrementally constructed, such that at any time t , we can obtain the current dataset D_t by a disjoint union of $O(\log t)$ subsets, each from v_t or a left ancestor of v_t in the interval tree. Consider each queried node v_i ($i \leq t$) where $D_t(v_i)$ denotes the set of items that node v_i stores at time t . This implies that a linear query $f(D_t)$ can be answered by computing the sum $\sum_i f(D_t(v_i))$ over queried nodes v_i , as specified in Section 3.1.2. Answering queries $\mathcal{F}$ on items stored by v_i at time t is a *deletion-only* problem: when $D(v_i)$ is first constructed at timestamp i , it has size $n(v_i) \leq n_i$ and no element is associated with deletion time. Then, items in $D(v_i)$ get deleted as time goes by.

Now we focus on the deletion-only problem at node v_i . To distinguish, we use $D(v_i)$ to denote the *initial state* of node v_i when it first gets constructed at timestamp i , and use $D_t(v_i)$ to denote the *remaining items* in node v_i at time t . Conceptually, we also consider a dynamic dataset $D_t^-(v_i)$, which consists of all the *deleted elements* from node v_i . Their sizes will be denoted as $n(v_i)$, $n_t(v_i)$ and $n_t^-(v_i)$ respectively. For example, $D(v_2) = \{a, b\}$ when constructed at time 2, whereas $D_5(v_2) = \{a\}$ and $D_5^-(v_2) = \{b\}$. Clearly at any time t , $D_t(v_i) \uplus D_t^-(v_i) = D(v_i)$ and therefore $n_t(v_i) + n_t^-(v_i) = n(v_i)$. We then have $n_t(v_i) \leq n_t$ and $n_t(v_i) \leq n(v_i) \leq n_i$, but there is no relationship between n_i and n_t .

A simple solution for this deletion-only problem is to first release a privatized $f(D(v_i))$ when initialized, and then run an insertion-only mechanism over the conceptual dataset $D_t^-(v_i)$ so that $f(D_t^-(v_i))$ can be obtained at any time. Using their difference, $f(D_t(v_i))$ can be answered. The problem here is that given a static mechanism $\mathcal{M}$ with error function $\alpha(|D|, \cdot)$, the initial $f(D(v_i))$ is answered with error $\alpha(n(v_i), \cdot)$. Although $n(v_i)$ is bounded by n_i , it has no relationship with the current data size n_t at query time. In particular, the initial data size $n(v_i)$ can be arbitrarily larger than its current size $n_t(v_i)$, which fails to achieve our target error bound $\alpha(n_t, \cdot)$.

To fix the problem, we ensure that no more than $n(v_i)/2$ items should be deleted so that at any time we can guarantee $n(v_i) = O(n_t)$. When half the items have been deleted from $D(v_i)$, we restart the process on a new $D(v_i)$ that consists of the remaining items with fresh privacy budgets.

There are still a few privacy-related issues with the above idea. First, we cannot restart when exactly half the items have been deleted, which would violate DP. Instead, we run a basic counting mechanism $\mathcal{M}_{cnt,ins}$ over the conceptual dataset for deletions to approximately keep track of the number of deletions; we show that such an approximation will only contribute an additive polylogarithmic error. Second, since we restart the process above multiple times, we need to allocate the privacy budget carefully using sequential composition. But the privacy degradation is only polylogarithmic since we have only $O(\log n(v_i)) = O(\log t)$ restarts with high probability. Finally, as the decision of restarting the process depends on random noises, the total number of restarts can not be fixed in advance. Since we must guarantee differential privacy for any possible instantiation, we need to allocate the privacy budget through a convergent sequence. Similar to [4], we allocate the privacy budgets proportional to $r^{-(1+\eta)}$ in round r for a small constant $\eta > 0$. The total privacy is then bounded regardless of the number of actual rounds, as $\sum_{r=1}^{\infty} r^{-(1+\eta)} < \frac{1}{\eta} + 1$. This incurs another logarithmic-factor degradation. Algorithm 1 details the steps we run at each node v in the online

interval tree. We present in Lemma 7 its accuracy guarantee, assuming each node is allocated with (ε, δ) -DP.

Algorithm 1: (ε, δ) -DP Algorithm at node $v = v_i$

Input: Fully-dynamic stream $(\dots, (s_t, x_t), \dots)$, constant η , privacy budget (ε, δ) , probability β , static $(\mathcal{M}_{\mathcal{F}})$ and insertion-only $(\mathcal{M}_{\mathcal{F}, ins})$ mechanisms for queries $\mathcal{F}$, continual counting mechanism $\mathcal{M}_{cnt, ins}$

Output: $\mathcal{F}(D_t(v_i))$ at any time $t \geq i$

```

1  $\theta \leftarrow \frac{\eta}{1+\eta}, r \leftarrow 1, (\varepsilon_1, \delta_1) \leftarrow (\frac{\theta}{4}\varepsilon, \frac{\theta}{2}\delta), \beta_1 \leftarrow \frac{\theta}{6}\beta;$  // Initialize
2  $D(v_i) \leftarrow$  All items in  $D_i$  inserted after the closest left-ancestor of  $v_i$  in online interval tree;
3  $\mathbf{D} \leftarrow D(v_i);$ 
4  $\hat{n} \leftarrow |\mathbf{D}| + \text{Lap}(1/\varepsilon_1);$ 
5 Release  $\mathcal{M}_{\mathcal{F}}(\mathbf{D})$  under  $(\varepsilon_1, \delta_1)$ -DP to answer  $\mathcal{F}(D_i(v_i));$ 
6 Initiate  $\mathcal{M}_{\mathcal{F}, ins}$  under  $(\varepsilon_1, \delta_1)$ -DP and  $\mathcal{M}_{cnt, ins}$  under  $\varepsilon_1$ -DP;
7 foreach  $t \leftarrow i + 1, i + 2, \dots$  do
8   if update  $(s_t, x_t) = (-, x)$  where  $x \in D(v_i)$  then
9     Augment the deletion time of  $x$  to  $(x, t);$ 
10    Feed an update  $(+, x)$  to  $\mathcal{M}_{\mathcal{F}, ins}$  and  $\mathcal{M}_{cnt, ins};$ 
11  else
12    Feed an update  $(\perp, x)$  to  $\mathcal{M}_{\mathcal{F}, ins}$  and  $\mathcal{M}_{cnt, ins};$ 
13   $\hat{n}^- \leftarrow$  Approximate number of deletions from  $\mathcal{M}_{cnt, ins};$ 
14   $\gamma \leftarrow$  Error bound  $\gamma_{cnt}(t; \varepsilon_r, \beta_r)$  in Corollary 1;
15  if  $\hat{n}^- > \hat{n}/2 + 2\gamma$  then // Restart
16     $r \leftarrow r + 1, (\varepsilon_r, \delta_r) \leftarrow (\frac{\theta}{4r^{1+\eta}}\varepsilon, \frac{\theta}{2r^{1+\eta}}\delta), \beta_r \leftarrow \frac{\theta}{6r^{1+\eta}}\beta;$ 
17     $\mathbf{D} \leftarrow D_t(v_i);$ 
18     $\hat{n} \leftarrow |\mathbf{D}| + \text{Lap}(1/\varepsilon_r);$ 
19    if  $\hat{n} < 2\gamma$  then // Terminate
20      Halt by answering 0 to all future  $\mathcal{F}(D_t(v_i));$ 
21    Release a new  $\mathcal{M}_{\mathcal{F}}(\mathbf{D})$  under  $(\varepsilon_r, \delta_r)$ -DP to answer  $\mathcal{F}(D_t(v_i));$ 
22    Restart  $\mathcal{M}_{\mathcal{F}, ins}$  under  $(\varepsilon_r, \delta_r)$ -DP and  $\mathcal{M}_{cnt, ins}$  under  $\varepsilon_r$ -DP;
23  else
24     $\mathcal{F}(\mathbf{D}) \leftarrow$  Query the current  $\mathcal{M}_{\mathcal{F}}(\mathbf{D});$ 
25     $\mathcal{F}(\mathbf{D}^-) \leftarrow$  Query the current  $\mathcal{M}_{\mathcal{F}, ins};$ 
26    Answer  $\mathcal{F}(D_t(v_i)) \leftarrow \mathcal{F}(\mathbf{D}) - \mathcal{F}(\mathbf{D}^-);$ 

```

Lemma 7. For each node v_i , Algorithm 1 is (ε, δ) -DP. Suppose there is a static mechanism $\mathcal{M}_{\mathcal{F}}$ for answering query class $\mathcal{F}$ is equipped with error function $\alpha^{(k)}(\varepsilon, \delta, \beta, |D|, |\mathcal{F}|, |\mathcal{X}|)$. Then at any time t , Algorithm 1 answers $\mathcal{F}(D_t(v_i))$ with error $\alpha^{(\log t)}\left(\frac{\varepsilon}{\tilde{O}(1)}, \frac{\delta}{\tilde{O}(1)}, \beta, n_t + \tilde{O}(1), |\mathcal{F}|, |\mathcal{X}|\right)$.

Proof. We use some extra notation in the proof for simplicity. As we focus on the deletion-only problem at node v_i , we denote $\mathbf{D} = D_{t_0}(v_i)$, $\mathbf{D}^- = \mathbf{D} - D_t(v_i)$, $\mathbf{n} = n_{t_0}(v_i)$ and $\mathbf{n}^- = \mathbf{n} - n_t(v_i)$, where t is the query time and t_0 is the beginning time of the current round. γ is the public error bound of basic counting on infinite streams given in Corollary 1, which only depends on parameters in the round r and the time t .

Privacy. Algorithm 1 uses four black-box mechanisms. In each round, $\hat{n}$ is released using the Laplace mechanism and $\mathcal{F}(\mathbf{D})$ is released using static mechanism $\mathcal{M}_{\mathcal{F}}$. In addition, two insertion-only mechanisms $\mathcal{M}_{\mathcal{F}, ins}$ and $\mathcal{M}_{cnt, ins}$ are used to track $\mathcal{F}$ and the basic counting query over the deletions $\mathbf{D}^-$. In any round r , the composition of these four mechanisms is $(4\varepsilon_r, 2\delta_r) = (\frac{\theta}{r^{1+\eta}}\varepsilon, \frac{\theta}{r^{1+\eta}}\delta)$ -DP. As we restart these mechanisms, they are sequentially composed, which guarantees the whole mechanism at node v_i is (ε, δ) -DP, since $\sum_{r=1}^{\infty} \frac{\theta}{r^{1+\eta}} < \frac{\eta}{1+\eta}(\frac{1}{\eta} + 1) = 1$. Note that the privacy guarantee holds regardless of the number of restarts.

Accuracy. Let random variable R denote the number of restarts before the mechanism terminates, we first bound R as follows. When a restart happens, we have $\hat{n}^- > \hat{n}/2 + 2\gamma$, where $\gamma = \tilde{O}(1)$. With probability $1 - 2\beta_r$, both $\hat{n}^-$ and $\hat{n}/2$ have error at most γ . Conditioned on this happening, we get $\mathbf{n}^- > \mathbf{n}/2$: at least half of the remaining items have been deleted from $\mathbf{D}$. As v_i was initialized with $n(v_i) \leq n_i \leq N_i \leq N_t$ items, with high probability this can happen at most $O(\log N_t)$ times before there are only $n_t(v_i) \leq \gamma$ items left. Still conditioned on the noise being small, line 19 becomes true and the algorithm halts by answering 0, which has error at most $O(\gamma)$ for any linear query. To conclude, conditioned on the events that in each round, the noises in counting are smaller than the bound γ , which happens with probability $1 - \sum_{r=1}^{\infty} 2\beta_r > 1 - \beta/3$, there can only be $O(\log N_t)$ rounds at time t , where N_t is the number of updates up till time t .

We next bound the error of $\mathcal{F}(D_t(v_i))$ in any round $r \leq R$, as a function of r and with probability $1 - 2\beta/3$. If the algorithm decides to restart at time t , an up-to-date dataset $D_t(v_i)$ is computed and a fresh $\mathcal{F}(D_t(v_i))$ is obtained from the static mechanism $\mathcal{M}_{\mathcal{F}}$ with privacy budget $(\varepsilon_r, \delta_r)$, whose error is $\alpha\left(\varepsilon_r, \delta_r, \frac{2\beta}{3}, n_t(v_i), \cdot\right) = \alpha(\varepsilon_r, \delta_r, \beta, n_t, \cdot)$ with probability $1 - 2\beta/3$. Otherwise (line 23), with probability $1 - 2\beta_r$ we have the actual number of deletions $\mathbf{n}^- \leq \mathbf{n}/2 + 4\gamma$. Namely the current data size is at least $n_t(v_i) = \mathbf{n} - \mathbf{n}^- \geq \mathbf{n}/2 - 4\gamma$. Conditioned on this, we calculate $\mathcal{F}(D_t(v_i))$ from the difference of $\mathcal{F}(\mathbf{D})$ and $\mathcal{F}(\mathbf{D}^-)$. Note that with probability $1 - \beta/6$, the error for $\mathcal{F}(\mathbf{D})$ from $\mathcal{M}_{\mathcal{F}}$ is $\alpha\left(\varepsilon_r, \delta_r, \frac{\beta}{6}, \mathbf{n}, \cdot\right)$ by definition; also with probability $1 - \beta/6$, the error for $\mathcal{F}(\mathbf{D}^-)$ from $\mathcal{M}_{\mathcal{F}, ins}$ is $\alpha^{(\log t)}\left(\frac{\varepsilon_r}{\log t}, \frac{\delta_r}{\log t}, \frac{\beta}{6}, \mathbf{n}^-, \cdot\right)$ by Lemma 4. But now both terms can be bounded by $\alpha^{(\log t)}\left(\frac{\varepsilon_r}{\log t}, \frac{\delta_r}{\log t}, \beta, n_t + \gamma, \cdot\right)$ with high probability, as $\mathbf{n}^- \leq \mathbf{n} \leq 2n_t(v_i) + 8\gamma = O(n_t + \gamma)$. This means their difference, $\mathcal{F}(D_t(v_i))$ has error $\alpha^{(\log t)}\left(\frac{\varepsilon_r}{\log t}, \frac{\delta_r}{\log t}, \beta, n_t + \gamma, \cdot\right)$ with probability at least $1 - 2\beta_r - 2\beta/6 \geq 1 - 2\beta/3$.

Finally we condition on the event that there are only $r = O(\log N_t)$ rounds to get rid of the dependency on r , which happens with all but $\beta/3$ probability. By Corollary 1, the counting error in round r at time t is $\gamma(t; \varepsilon_r, \beta_r) = O\left(\frac{\log^{1.5} t}{\varepsilon_r} \sqrt{\log \frac{1}{\beta_r}} + \frac{\log t}{\varepsilon_r} \cdot \log \frac{1}{\beta_r}\right) = O\left(\frac{\log^{1.5} t}{\varepsilon_r} \cdot \log \frac{1}{\beta_r}\right)$. Therefore the error for $\mathcal{F}(D_t(v_i))$ is (with probability $1 - \beta$)

$$\alpha^{(\log t)}\left(\frac{\varepsilon}{(\log^{1+\eta} N_t) \log t}, \frac{\delta}{(\log^{1+\eta} N_t) \log t}, \beta, n_t + \frac{(\log^{1+\eta} N_t) \log^{1.5} t}{\varepsilon} \log\left(\frac{\log N_t}{\beta}\right), |\mathcal{F}|, |\mathcal{X}|\right).$$

□

3.3 Full Algorithm

Lemma 7 assumes each node is under (ε, δ) -DP, which we cannot afford since we have an online interval tree of depth $\log t$. Since the tree size grows with t and can be infinite, we allocate $(\varepsilon(v), \delta(v))$ proportional to $1/\ell^{1+\eta}$ to a node v at level ℓ in the online interval tree, so that the composition of mechanisms at all the nodes still satisfies (ε, δ) -DP at any time t . Since each item will only affect one single node in each level, nodes at the same level enjoy parallel composition. The error of the final sum is then decomposed into $\log t$ online interval tree nodes, where the error of each node is given by Lemma 7 but under $(\varepsilon(v), \delta(v))$ -DP.

To provide an error guarantee to the final results, note that it is the disjoint union of $(\log^2 t)$ static mechanisms. This is because each of the $(\log t)$ queried online interval tree nodes runs an insertion-only mechanism to support querying the deleted elements, the error of each one is from the error of $(\log t)$ static mechanisms. We are left with analyzing the error of each building block: the static mechanisms. Their error depends on the privacy budgets (ε', δ') allocated to it. In our construction, it has a polylogarithmic degradation compared to the (ε, δ) for the whole mechanism. In particular, there are 3 factors that accounts for the allocation of privacy budgets.

1. We have an online interval tree of height $(\log t)$. While nodes on the same level enjoy parallel composition, it is possible that the change of one timestamp affects multiple tree nodes on the root-to-node path corresponding to this timestamp. Therefore, we must allocate privacy budget with sequential composition. Moreover, the tree can grow infinitely. So

instead of an even allocation, we apply a convergent sequence to divide the budgets, which causes a $\log^{1+\eta} t$ overhead to the worst building block at time t .

2. At each online interval tree node, we restart the 4 mechanisms several times. Since this may repeatedly reveal information of the same entry, we also need to divide the privacy budget accordingly. Again, we allocate the budgets in each round using a convergent sequence again to make the DP guarantee independent of the number of restarts. With high probability, no queried node will restart more than $\log N_t$ times, so the minimum privacy budget in any round is a $1/(\log^{1+\eta} N_t)$ fraction of the budget allocated to the node.
3. Finally, in each round of each node, the insertion-only mechanisms $\mathcal{M}_{cnt,ins}$ and $\mathcal{M}_{\mathcal{F},ins}$ are built from static mechanisms, each receiving a $1/(\log t)$ fraction of the budget.

Putting things together, we arrive at the main result of this paper. For the running time, observe that the mechanism run at each node of $\mathcal{T}$ is dominated by the insertion-only mechanism from Lemma 4 that tracks the deletions. Since there can be $O(\log t)$ restarts, and the total running time over all nodes in the same level is $O(\log t) \cdot \text{time}(N_t)$. Summing over all $O(\log t)$ levels, we obtain the running time stated.

Theorem 8. *If there is a static DP mechanism with error function $\alpha^{(k)}(\varepsilon, \delta, \beta, |D|, |\mathcal{F}|, |\mathcal{X}|)$ for a set of union-preserving queries $\mathcal{F}$, then there is an (ε, δ) -DP mechanism $\mathcal{M}_{dyn}$ for fully-dynamic streams, so that at any time t , it answers queries $\mathcal{F}$ on D_t with error*

$$\zeta(t; \varepsilon, \delta, \beta, n_t, |\mathcal{F}|, |\mathcal{X}|) = \alpha^{(\log^2 t)} \left(\frac{\varepsilon}{\log^{3+2\eta} t}, \frac{\delta}{\log^{3+2\eta} t}, \beta, n_t + \frac{\log^{3.5+2\eta} t}{\varepsilon} \log \left(\frac{\log t}{\beta} \right), |\mathcal{F}|, |\mathcal{X}| \right),$$

for any constant $\eta > 0$. The running time of $\mathcal{M}_{dyn}$ up to time t is $O(\log^2 t) \cdot \text{time}(N_t)$.

Plugging in PMW as the static DP mechanism, we obtain a fully dynamic algorithm for answering a set of arbitrary linear queries with the following error bounds:

Corollary 2. *Given a set of linear queries $\mathcal{F}$, there is an (ε, δ) -DP mechanism $\mathcal{M}_{\mathcal{F},dyn}$ that at any time t , answers any query $f \in \mathcal{F}$ on a fully-dynamic stream with error*

$$\zeta_{\mathcal{F}}(t; \varepsilon, \delta, \beta, n_t, |\mathcal{F}|, |\mathcal{X}|) = \alpha_{\text{PMW}}^{(\tilde{O}(1))} \left(\frac{\varepsilon}{\tilde{O}(1)}, \frac{\delta}{\tilde{O}(1)}, \beta, n_t + \tilde{O}(1), |\mathcal{F}|, |\mathcal{X}| \right) = \begin{cases} \tilde{O}(\sqrt{n_t}), & \delta > 0, \\ \tilde{O}(n_t^{2/3}), & \delta = 0. \end{cases}$$

Our mechanism only has a polynomial dependency on the data size n_t at time t , matching results in the static setting, whereas a straightforward extension of insertion-only mechanisms [9, 4] will have a polynomial dependency on the number of updates $N_t \gg n_t$.

4 Limitations

While polylogarithmic factors are often neglected in theoretical studies, they still present significant overhead in practice, which limits the practicality of the algorithms proposed in this paper. How to reduce this overhead remains an interesting problem for further investigation. A possible future direction is to consider white-box mechanisms that improve the accuracy (though they can only reduce the polylogarithmic factors) while being practical. It is also interesting to study how to answer non-union-preserving queries (e.g. distinct count) accurately under the streaming DP setting.

Acknowledgments and Disclosure of Funding

This work is supported by HKRGC under grants 16205422, 16204223, and 16203924; and by the Start-up Research Fund of Southeast University under grant RF1028625150. We would also like to thank the anonymous reviewers who have made valuable suggestions on improving the presentation of the paper.

References

- [1] A. Blum, K. Ligett, and A. Roth. A learning theory approach to noninteractive database privacy. *J. ACM*, 60(2):12:1–12:25, 2013.
- [2] M. Bun and T. Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Proc. Theory of Cryptography, TCC*, volume 9985, pages 635–658, 2016.
- [3] A. R. Cardoso and R. Rogers. Differentially private histograms under continual observation: Streaming selection into the unknown. In *AISTATS*, volume 151 of *Proceedings of Machine Learning Research*, pages 2397–2419. PMLR, 2022.
- [4] T. H. Chan, E. Shi, and D. Song. Private and continual release of statistics. In *Proc. Automata, Languages and Programming, ICALP*, volume 6199, pages 405–417. Springer, 2010.
- [5] G. Cormode and K. Yi. *Small summaries for big data*. Cambridge University Press, 2020.
- [6] R. Cummings, S. Krehbiel, K. A. Lai, and U. T. Tantipongpipat. Differential privacy for growing databases. In *Proc. Advances in Neural Information Processing Systems, NeurIPS*, pages 8878–8887, 2018.
- [7] M. de Berg, O. Cheong, M. J. van Kreveld, and M. H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008. ISBN 9783540779735.
- [8] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.
- [9] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *Proc. ACM Symposium on Theory of Computing, STOC*, pages 715–724. ACM, 2010.
- [10] C. Dwork, G. N. Rothblum, and S. P. Vadhan. Boosting and differential privacy. In *IEEE Symposium on Foundations of Computer Science, FOCS*, pages 51–60. IEEE Computer Society, 2010.
- [11] C. Dwork, M. Naor, O. Reingold, and G. N. Rothblum. Pure differential privacy for rectangle queries via private partitions. In *Proc. Advances in Cryptology, ASIACRYPT*, volume 9453, pages 735–751. Springer, 2015.
- [12] H. Fichtenberger, M. Henzinger, and L. Ost. Differentially private algorithms for graphs under continual observation. In *ESA*, volume 204 of *LIPIcs*, pages 42:1–42:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.
- [13] H. Fichtenberger, M. Henzinger, and J. Upadhyay. Constant matters: Fine-grained complexity of differentially private continual observation using completely bounded norms. In *Proc. International Conference on Machine Learning, ICML*, 2023.
- [14] A. Ganesh and J. Zhao. Privately answering counting queries with generalized gaussian mechanisms. In *Symposium on Foundations of Responsible Computing, FORC*, volume 192 of *LIPIcs*, pages 1:1–1:18, 2021.
- [15] M. Hardt and G. N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *IEEE Symposium on Foundations of Computer Science, FOCS*, pages 61–70. IEEE Computer Society, 2010.
- [16] M. Hardt and K. Talwar. On the geometry of differential privacy. In *Proc. ACM Symposium on Theory of Computing, STOC*, pages 705–714. ACM, 2010.
- [17] M. Hardt, K. Ligett, and F. McSherry. A simple and practical algorithm for differentially private data release. In *Proc. Advances in Neural Information Processing Systems, NeurIPS*, pages 2348–2356, 2012.
- [18] M. Henzinger, J. Upadhyay, and S. Upadhyay. Almost tight error bounds on differentially private continual counting. In *Proc. ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 5003–5039. SIAM, 2023.
- [19] P. Jain, I. Kalemaj, S. Raskhodnikova, S. Sivakumar, and A. D. Smith. Counting distinct elements in the turnstile model with differential privacy under continual observation. In *NeurIPS*, 2023.
- [20] P. Jain, A. D. Smith, and C. Wagaman. Time-aware projections: Truly node-private graph statistics under continual observation. In *SP*, pages 127–145. IEEE, 2024.

- [21] P. Kairouz, S. Oh, and P. Viswanath. The composition theorem for differential privacy. *IEEE Trans. Inf. Theory*, 63(6):4037–4049, 2017.
- [22] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *Proc. ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*, pages 123–134. ACM, 2010.
- [23] C. Li, G. Miklau, M. Hay, A. McGregor, and V. Rastogi. The matrix mechanism: optimizing linear counting queries under differential privacy. *VLDB J.*, 24(6):757–781, 2015.
- [24] F. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proc. ACM SIGMOD International Conference on Management of Data, SIGMOD*, pages 19–30. ACM, 2009.
- [25] S. Muthukrishnan. Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 1(2), 2005.
- [26] S. Muthukrishnan and A. Nikolov. Optimal private halfspace counting via discrepancy. In *Proc. Symposium on Theory of Computing Conference, STOC*, pages 1285–1292. ACM, 2012.
- [27] S. Raskhodnikova and T. A. Steiner. Fully dynamic algorithms for graph databases with edge differential privacy. *Proc. ACM Manag. Data*, 3(2):99:1–99:28, 2025.
- [28] S. Song, S. Little, S. Mehta, S. A. Vinterbo, and K. Chaudhuri. Differentially private continual release of graph statistics. *CoRR*, abs/1809.02575, 2018.
- [29] T. Steinke and J. R. Ullman. Between pure and approximate differential privacy. *J. Priv. Confidentiality*, 7(2), 2016.
- [30] S. P. Vadhan. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*, pages 347–450. Springer International Publishing, 2017.
- [31] R. Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge University Press, 2018.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes]" is generally preferable to "[No]", it is perfectly acceptable to answer "[No]" provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No]" or "[NA]" is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading “NeurIPS Paper Checklist”,**
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers.**

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope?

Answer: [Yes]

Justification: In both the abstract and the introduction, we stated that this paper presents black-box algorithms for DP queries on fully-dynamic streams with a polylogarithmic degradation in the utility compared to static ones.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: We have discussed the limitations in Section 4.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We stated the assumptions in Section 2, and presented our results with proofs in Section 3.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [NA]

Justification: This is a theoretical paper that does not include experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [NA]

Justification: This is a theoretical paper that does not include experiments.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [NA]

Justification: This is a theoretical paper that does not include experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [NA]

Justification: This is a theoretical paper that does not include experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [NA]

Justification: This is a theoretical paper that does not include experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: This paper confirms with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: This paper studies DP query answering from a theoretical perspective, which should not have a societal impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This paper poses no such risks since it does not involve specific data or model.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [NA]

Justification: This paper does not use existing assets.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: This paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in the research does not involve LLMs.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

A DP Mechanisms for Linear Queries: A Review

In this section, we present some important DP mechanisms for linear queries and their error bounds. The analysis is similar to that in [30], but we clarify the dependency on β , which will be needed for analyzing the simultaneous error in the streaming setting. Note that a linear query has sensitivity 1 by definition.

Laplace Mechanism. When $\mathcal{F} = \{f\}$ is a single query, the Laplace mechanism that outputs $\mathcal{M}_{\text{Lap}}(D) = f(D) + \text{Lap}(\frac{1}{\varepsilon})$ has error $\alpha_{\text{Lap}}(\varepsilon, \beta) = \frac{1}{\varepsilon} \ln \frac{1}{\beta}$. When $\mathcal{F}$ contains multiple queries, we may add $\text{Lap}(\frac{|\mathcal{F}|}{\varepsilon})$ noise to each query result and apply basic composition to guarantee ε -DP of the whole mechanism. To translate it into an error bound, we bound the failure probability of each noise by $\frac{\beta}{|\mathcal{F}|}$, so that a union bound will bring the total failure probability to β . A similar argument can be made using advanced composition. To conclude, answering a set of queries $\mathcal{F}$ using Laplace mechanism achieves error (for $\delta \geq 0$)

$$\alpha_{\text{Lap}}(\varepsilon, \delta, \beta, |\mathcal{F}|) = \begin{cases} O\left(\frac{|\mathcal{F}|}{\varepsilon} \log \frac{|\mathcal{F}|}{\beta}\right) & , \delta \leq e^{-\Omega(|\mathcal{F}|)}; \\ O\left(\frac{\sqrt{|\mathcal{F}| \log \frac{1}{\delta}}}{\varepsilon} \log \frac{|\mathcal{F}|}{\beta}\right) & , \delta \geq e^{-O(|\mathcal{F}|)}. \end{cases}$$

Gaussian Mechanism. Similar to the Laplace mechanism, the Gaussian mechanism protects (ε, δ) -DP of query f by outputting $\mathcal{M}_{\text{Gauss}}(D) = f(D) + \mathcal{N}\left(0, \frac{2}{\varepsilon^2} \ln \frac{1.25}{\delta}\right)$. Its error is $\alpha_{\text{Gauss}}(\varepsilon, \delta, \beta) = \frac{2}{\varepsilon} \sqrt{\ln \frac{1.25}{\delta} \ln \frac{2}{\beta}}$. When composing multiple Gaussian mechanisms that each answers a query from $\mathcal{F}$, zCDP composition [2] can be applied, which shows adding $\mathcal{N}\left(0, O\left(\frac{|\mathcal{F}|}{\varepsilon^2} \log \frac{1}{\delta}\right)\right)$ noise to each query suffices to protect (ε, δ) -DP of the whole mechanism. Therefore answering a set of queries $\mathcal{F}$ using Gaussian mechanism achieves error (for $\delta > 0$)

$$\alpha_{\text{Gauss}}(\varepsilon, \delta, \beta, |\mathcal{F}|) = O\left(\frac{\sqrt{|\mathcal{F}| \log \frac{1}{\delta}}}{\varepsilon} \sqrt{\log \frac{|\mathcal{F}|}{\beta}}\right).$$

Private Multiplicative Weights. When there are many queries $|\mathcal{F}| = \Omega(|D|)$, composing individual mechanisms has error polynomial in $|\mathcal{F}|$, thus also in $|D|$. The Private Multiplicative Weights mechanism [15, 17] performs better in this case. The following error bound is presented in [17, 8].

$$\alpha_{\text{PMW}} = \begin{cases} O\left(\left(\frac{|D|^2 \log |\mathcal{X}| \log(|\mathcal{F}|/\beta)}{\varepsilon}\right)^{\frac{1}{3}}\right) & , \delta = 0; \\ O\left(\left(\frac{|D| \sqrt{\log |\mathcal{X}| \log(1/\delta) \log(|\mathcal{F}|/\beta)}}{\varepsilon}\right)^{\frac{1}{2}}\right) & , \delta > 0. \end{cases}$$

Apart from mechanisms mentioned above, there are other private mechanisms for linear queries. For example, the optimal composition [21] can be used in place of basic or advanced composition to provide a better allocation of privacy budget, yet computing it is costly. The $\log |\mathcal{F}|$ factor is removable for Laplace mechanism [29] and almost removable for Gaussian mechanism [14]. Under pure-DP, SmallDB [1] has asymptotically the same error as PMW, but its running time is prohibitive. The Matrix mechanism [22, 23] exploits structural properties within the query set $\mathcal{F}$ and works well in practice. But it does not have a closed-form error bound for general queries, and finding an optimal querying strategy is time-consuming.

In general, the best mechanism is related to the hereditary discrepancy [16, 26] of the set of queries. For example, for d -dimensional halfspace counting queries, [26] has error $\tilde{O}(|D|^{\frac{1}{2} - \frac{1}{2d}})$. In this paper we use α as a function of $\varepsilon, \delta, \beta$, and possibly $|D|, |\mathcal{F}|, |\mathcal{X}|$ to denote the error of any mechanism answering linear queries on *static* datasets, without detailing the best mechanism under a specific setting and choice of the parameters. Since our paper takes a black-box approach, all these algorithms can be plugged into our framework so as to support dynamic data, while incurring a polylogarithmic-factor degradation.

B Error Bounds under Disjoint Union

Note that $\alpha^{(k)}$ denotes the error of the sum of k mechanisms, each of which having error α under the same parameter settings. It always holds that $\alpha^{(k)}(\cdot, \beta) \leq k \cdot \alpha(\cdot, \frac{\beta}{k})$ by union bound. In this section, we show cases where $\alpha^{(k)}$ can be tightened for specific mechanisms. As our running example, consider $\alpha_{\text{Lap}}(\varepsilon, \beta) = \frac{1}{\varepsilon} \ln \frac{1}{\beta}$. The union bound reduction above gives

$$\alpha_{\text{Lap}}^{(k)}(\varepsilon, \beta) \leq \frac{k}{\varepsilon} \ln \frac{k}{\beta}$$

to bound the error of summing k Laplace noises $\text{Lap}(\frac{1}{\varepsilon})$. Next we show how this can be tightened.

B.1 Unbiasedness

If a mechanism $\mathcal{M}$ is *unbiased* with error α , naturally the error only scales with $\sqrt{k}$. To see why this is true, we can argue that with all but $\frac{\beta}{2}$ probability, individual mechanisms have their errors bounded by $\alpha(\cdot, \frac{\beta}{2k})$ simultaneously. Conditioned on this happening, apply Hoeffding's inequality with the remaining $\frac{\beta}{2}$ probability, we get

$$\alpha_{\text{Unbiased}}^{(k)}(\cdot, \beta) \leq \sqrt{2k \ln \frac{4}{\beta}} \cdot \alpha\left(\cdot, \frac{\beta}{2k}\right).$$

This can be applied to the Laplace mechanism to get

$$\alpha_{\text{Lap}}^{(k)}(\varepsilon, \beta) = O\left(\frac{\sqrt{k}}{\varepsilon} \sqrt{\log \frac{1}{\beta}} \log \frac{k}{\beta}\right).$$

B.2 Concentration Bounds

For specific distributions like Laplace (sub-exponential) and Gaussian (sub-gaussian), the concentration bounds are usually tighter than using unbiasedness only. It helps save the $\log k$ factor from union bound. Consider the Laplace mechanism, note that the $\text{Lap}(\frac{1}{\varepsilon})$ random variable is sub-exponential with norm $\|\text{Lap}(\frac{1}{\varepsilon})\|_{\Psi_1} = \frac{2}{\varepsilon}$. We can then apply Bernstein's inequality [31].

Lemma 9 (Bernstein's inequality). *Let $X_1, \dots, X_k$ be i.i.d. zero-mean sub-exponential random variables with norm Ψ_1 . There is an absolute constant c so that for any $t \geq 0$,*

$$\Pr\left[\left|\sum_{i=1}^k X_i\right| > t\right] \leq 2 \exp\left[-c \min\left\{\frac{t^2}{k\Psi_1^2}, \frac{t}{\Psi_1}\right\}\right].$$

We therefore conclude the Laplace mechanism has error function

$$\alpha_{\text{Lap}}^{(k)}(\varepsilon, \beta) = O\left(\frac{1}{\varepsilon} \sqrt{\left(k + \log \frac{1}{\beta}\right) \cdot \log \frac{1}{\beta}}\right). \quad (2)$$

As another example, for the Gaussian mechanism, the sum of k Gaussian noises is still a Gaussian noise with its variance scaled up by k , thus the disjoint union of k Gaussian mechanisms has error

$$\alpha_{\text{Gauss}}^{(k)}(\varepsilon, \delta, \beta) = O\left(\frac{1}{\varepsilon} \sqrt{k \log \frac{1}{\delta} \log \frac{1}{\beta}}\right).$$

For the PMW mechanism, the union bound remains the best we know, that is

$$\begin{aligned} \alpha_{\text{PMW}}^{(k)} &= \begin{cases} O\left(k \cdot \left(\frac{|D|^2 \log |\mathcal{X}| \log(k|\mathcal{F}|/\beta)}{\varepsilon}\right)^{\frac{1}{3}}\right) & , \delta = 0; \\ O\left(k \cdot \left(\frac{|D| \sqrt{\log |\mathcal{X}| \log(1/\delta)} \log(k|\mathcal{F}|/\beta)}{\varepsilon}\right)^{\frac{1}{2}}\right) & , \delta > 0. \end{cases} \\ &= \begin{cases} \tilde{O}\left(k \cdot |D|^{\frac{2}{3}}\right) & , \delta = 0; \\ \tilde{O}\left(k \cdot |D|^{\frac{1}{2}}\right) & , \delta > 0. \end{cases} \end{aligned}$$

C Related Work for DP under Continual Observation

In the Binary Tree mechanism [9], each timestamp is treated as a leaf node in the binary tree, and the mechanism privately releases the count of each tree node. If the stream has bounded length T , then each node receives a $\frac{1}{\log T}$ fraction of the privacy budget, and the final count consists of counts from $O(\log t)$ tree nodes. Using our notation, the mechanism has error $\alpha_{\text{Lap}}^{(\log t)}(\frac{\varepsilon}{\log T}, \beta)$ for basic counting. Chan et al. [4] first showed that allocating the privacy budget through a convergent sequence will allow the Binary Tree mechanism to also work for infinite streams, where the error is worse by a logarithmic factor as $\alpha_{\text{Lap}}^{(\log t)}(\frac{\varepsilon}{\log^{1+\eta} t}, \beta)$ for constant $\eta > 0$. They further proposed a hybrid mechanism to show that $\alpha_{\text{Lap}}^{(\log t)}(\frac{\varepsilon}{\log t}, \beta)$ can be achieved by separately releasing the counts for $[1, 2), [2, 4), [4, 8), \dots$. Using the $\alpha_{\text{Lap}}^{(k)}$ function in Equation (2) above, we obtain the error

$$\alpha_{\text{Lap}}^{(\log t)}\left(\frac{\varepsilon}{\log t}, \beta\right) = O\left(\frac{\log^{1.5} t}{\varepsilon} \sqrt{\log \frac{1}{\beta}} + \frac{\log t}{\varepsilon} \cdot \log \frac{1}{\beta}\right).$$

for continual counting, which was presented in Corollary 1 as our building block. Note that this is a *per-query* bound: with constant probability, a query at any time t can be answered with error $O(\frac{1}{\varepsilon} \log^{1.5} t)$.

Alternatively, if we replace β with β/T where T is an upper bound on t , we obtain a bound that holds *simultaneously* for all queries $1 \leq t \leq T$: with constant probability, any query can be answered with error $O(\frac{1}{\varepsilon} \log^2 T)$. The *simultaneous* bound given in [9] was $O(\frac{1}{\varepsilon} \log^{2.5} T)$ using unbiasedness but not concentration, which was corrected in [11]. In [4], Chan et al. presented the $\alpha^{(k)}$ function as $\alpha_{\text{Lap}}^{(k)}(\varepsilon, \beta) = O\left(\frac{\sqrt{k}}{\varepsilon} \log \frac{1}{\beta}\right)$, and thus

$$\alpha_{\text{Lap}}^{(\log t)}\left(\frac{\varepsilon}{\log t}, \beta\right) = O\left(\frac{\log^{1.5} t}{\varepsilon} \log \frac{1}{\beta}\right).$$

While the per-query bound is the same, for the simultaneous bound, this only gives $O(\frac{1}{\varepsilon} \log^{2.5} T)$, whereas our tighter bound gives $O(\frac{1}{\varepsilon} \log^2 T)$. The same simultaneous bound was presented in [13, 18] with improved constants using the Matrix Mechanism [22, 23] as building block. For sparse finite streams, [11] achieves $O(\frac{\log^{1.5} N_t}{\varepsilon} \log \frac{1}{\beta} + \frac{1}{\varepsilon} \log \frac{T}{\beta})$ per-query error for basic counting, which has an asymptotic improvement when $N_t = t^{o(1)}$.

We briefly mention some other work under similar settings but are less related. [3] studies the DP histogram problem in the streaming setting, which is a special case of linear queries. Their main contribution is when the universe $\mathcal{X}$ or the sensitivity of the queries are unbounded. Otherwise, the algorithm is equivalent to Corollary 1. [19] studies the DP distinct counting problem under the turnstile (fully-dynamic) model. Distinct counting queries are known to be non-linear and not union-preserving, which is separate from the interest of this paper. There are also existing work that study graph queries under the continual observation model of differential privacy, e.g. [28, 12, 20, 27].