Differentially Private L_2-Heavy Hitters in the Sliding Window Model (Full Version)

Anonymous submission

Abstract

The data management of large companies often prioritize more recent data, as a source of higher accuracy prediction than outdated data. For example, the Facebook data policy retains user search histories for 6 months while the Google data retention policy states that browser information may be stored for up to 9 months. These policies are captured by the sliding window model, in which only the most recent W statistics form the underlying dataset.

In this paper, we consider the problem of privately releasing the L_2-heavy hitters in the sliding window model, which include L_p-heavy hitters for $p \leq 2$ and in some sense are the strongest possible guarantees that can be achieved using polylogarithmic space, but cannot be handled by existing techniques due to the sub-additivity of the L_2 norm. Moreover, existing non-private sliding window algorithms use the smooth histogram framework, which has high sensitivity.

To overcome these barriers, we introduce the first differentially private algorithm for L_2-heavy hitters in the sliding window model by initiating a number of L_2-heavy hitter algorithms across the stream with significantly lower threshold. Similarly, we augment the algorithms with an approximate frequency tracking algorithm with significantly higher accuracy. We then use smooth sensitivity and statistical distance arguments to show that we can add noise proportional to an estimation of the L_2 norm. To the best of our knowledge, our techniques are the first to privately release statistics that are related to a sub-additive function in the sliding window model, and may be of independent interest to future differentially private algorithmic design in the sliding window model.

1 Introduction

Differential privacy [Dwo06, DMNS16] has emerged as the standard for privacy in the both the research and industrial communities. For example, Google Chrome uses RAPPOR [EPK14] to collect user statistics such as the default homepage of the browser or the default search engine, etc., Samsung proposed a similar mechanism to collect numerical answers such as the time of usage and battery volume [NXY+16], and Apple uses a differentially private method [Gre16] to generate predictions of spellings.

The age of collected data can significantly impact its relevance to predicting future patterns, as the behavior of groups or individuals may significantly change over time due to either cyclical, temporary, or permanent change. Indeed, recent data is often a more accurate predictor than older data across multiple sources of big data, such as stock markets or Census data, a concept which is often reflected through the data management of large companies. For example, the Facebook data policy [Fac] retains user search histories for 6 months, the Apple differential privacy [Upa19] states that collected data is retained for 3 months, the Google data retention policy states that browser information may be stored for up to 9 months [Goo], and more generally, large data collection

agencies often perform analysis and release statistics on time-bounded data. However, since large data collection agencies often manage highly sensitive data, the statistics must be released in a way that does not compromise privacy. Thus in this paper, we study the (event-level) differentially private release of statistics of time-bounded data that only use space sublinear in the size of the data.

Definition 1.1 (Differential privacy [DMNS16])**.** *Given $\varepsilon > 0$ and $\delta \in (0, 1)$, a randomized algorithm $\mathcal{A}$ operating on datastreams is (ε, δ)-differentially private if, for every pair of neighboring datasets $\mathfrak{S}$ and $\mathfrak{S}'$ and for all sets E of possible outputs, we have,*

$$\mathbf{Pr}\left[\mathcal{A}(\mathfrak{S}) \in E\right] \leq e^{\varepsilon} \cdot \mathbf{Pr}\left[\mathcal{A}(\mathfrak{S}') \in E\right] + \delta.$$

In the popular streaming model of computation, elements of an underlying dataset arrive one-by-one but the entire dataset is considered too large to store; thus algorithms are restricted to using space sublinear in the size of the data. Although the streaming model provides a theoretical means to handle big data and has been studied thoroughly for applications in privacy-preserving data analysis, e.g., [MMNW11, BBDS12, JRUW20, HKM+20, HQYC21], it does not properly capture the ability to prioritize more recent data, which is a desirable quality for data summarization. The time decay model [CS06, KP08, SYC18, BLUZ19] emphasizes more recent data by assigning a polynomially decaying or exponentially decaying weight to "older" data points, but these functions cannot capture the zero-one property when data older than a certain age is completely deleted.

The sliding window model. By contrast, the *sliding window model* takes a large data stream as an input and only focuses on the updates past a certain point in time by implicitly defining the underlying dataset through the most recent W updates of the stream, where $W > 0$ is the window parameter. Specifically, given a stream $u_1, \ldots, u_m$ such that $u_i \in [n]$ for all $i \in [m]$ and a parameter $W > 0$ that we assume satisfies $W \leq m$ without loss of generality, the underlying dataset is a frequency vector $f \in \mathbb{R}^n$ induced by the last W updates of the stream $u_{m-W+1}, \ldots, u_m$ so that

$$f_k = |\{i : u_i = k\}|,$$

for all $k \in [n]$. Then the goal is to output a private approximation to the frequency f_k of each heavy-hitter, i.e., the indices $k \in [n]$ for which $f_k \geq \alpha L_p(f)$, which denotes the L_p norm of f for a parameter $p \geq 1$:

$$L_p(f) = \|f\|_p = \left(\sum_{i=1}^{n} f_i^p\right)^{1/p}.$$

In this case, we say that streams $\mathfrak{S}$ and $\mathfrak{S}'$ are neighboring if there exists a single update $i \in [m]$ such that $u_i \neq u_i'$, where $u_1, \ldots, u_m$ are the updates of $\mathfrak{S}$ and $u_1', \ldots, u_m'$ are the updates of $\mathfrak{S}'$.

Note that if k is an L_1-heavy hitter, i.e., a heavy-hitter with respect to $L_1(f)$, then $f_k \geq \alpha L_1(f)$ so that

$$f_k \geq \alpha \left(\sum_{i=1}^{n} f_i\right) \geq \alpha \left(\sum_{i=1}^{n} f_i^2\right)^{1/2},$$

and k is also an L_2-heavy hitter. Thus, any L_2-heavy hitter algorithm will also report the L_1-heavy hitters, but the converse is not always true. Indeed, for the Yahoo! password frequency corpus [BDB16] ($n \approx 70$ million) with heavy-hitter threshold $\alpha = \frac{1}{500}$ there were $3,972$ L_2-heavy

hitters, but only one L_1-heavy hitter. On the other hand, finding L_p-heavy hitters for $p > 2$ requires $\Omega(n^{1-2/p})$ space [CKS03, BJKS04], so in some sense, the L_2-heavy hitters are the best we can hope to find using polylogarithmic space. Although there is a large and active line of work in the sliding window model [DGIM02, BO07, BGO14, BLLM16, BGL+18, BDM+20, BEL+20, WZ21, BWZ21, JWZ21], there is surprisingly little work in the sliding window model that considers differential privacy [Upa19, UU21].

1.1 Our Contributions

In this paper, we consider the problem of privately releasing approximate frequencies for the heavy-hitters in a dataset defined by the sliding window model. We give the first differentially private algorithm for approximating the frequencies of the L_2-heavy hitters in the sliding window model.

Theorem 1.2. *For any $\alpha \in (0,1), c > 0$, window parameter W on a stream of length m that induces a frequency vector $f \in \mathbb{R}^n$ in the sliding window model, and privacy parameter $\varepsilon > \frac{1000 \log m}{\alpha^3 \sqrt{W}}$, there exists an algorithm such that:*

(1) (Privacy) The algorithm is (ε, δ)-differentially private for $\delta = \frac{1}{m^c}$.

(2) (Heavy-hitters) With probability at least $1 - \frac{1}{m^c}$, the algorithm outputs a list $\mathcal{L}$ such that $k \in \mathcal{L}$ for each $k \in [n]$ with $f_k \geq \alpha\, L_2(f)$ and $j \notin \mathcal{L}$ for each $j \in [n]$ with $f_j \leq \frac{\alpha}{2}\, L_2(f)$.

(3) (Accuracy) With probability at least $1 - \frac{1}{m^c}$, we simultaneously have $|f_k - \widetilde{f}_k| \leq \frac{\alpha}{4}\, L_2(f)$ for all $k \in \mathcal{L}$, where $\widetilde{f}_k$ denotes the noisy approximation of f_k output by the algorithm.

(4) (Complexity) The algorithm uses $\mathcal{O}\left(\frac{\log^7 m}{\alpha^6 \eta^4}\right)$ bits of space and $\mathcal{O}\left(\frac{\log^4 m}{\alpha^3 \eta^4}\right)$ operations per update where $\eta = \max\{1, \varepsilon\}$.

Along the way, we develop techniques for handling differentially private heavy-hitter algorithms in the sliding window model that may be of independent interest. In particular, we also use our techniques to obtain an L_1-heavy hitter algorithm for the sliding window model that guarantees *pure* differential privacy. Finally, we give an algorithm for continual release of L_1 and L_2-heavy hitters in the sliding window model that has additive error $\frac{\alpha\sqrt{W}}{2}$ for each estimated heavy-hitter frequency and preserves pure differential privacy, building on a line of work [CLSX12, Upa19, HQYC21] for continual release. By comparison, the algorithm of [Upa19] only guarantees $\mathcal{O}\left(W^{3/4}\right)$ additive error while the algorithm of [HQYC21] gives (ε, δ)-differential privacy. We remark that since $\sqrt{W} \leq L_2(t - W + 1 : t)$ for any $t \in [m]$, where $L_2(t - W + 1 : t)$ denotes the L_2 norm of the sliding window between times $t - W + 1$ and t, then our improvements over [Upa19] for the continual release of L_1-heavy hitters actually also resolve the problem of continual release of L_2-heavy hitters. Nevertheless, the approach is somewhat standard and thus we defer discussion to the appendix.

1.2 Related Work

Dynamic structures vs. linear sketching. Non-private algorithms in the streaming model generally follow one of two main approaches. The first main approach is the transformation from static data structures to dynamic structures using the framework of [BS80]. Although the approach has been a useful tool for many applications [DNP+10, CSS11, CLSX12, LMWY20], it does provide a mechanism to handle the implicit deletion of updates induced by the sliding window model. The

second main approach is the use of linear sketching [BBDS12, BS15, BNS19, BNST20, HQYC21], where the data x is multiplied by a random matrix A to create a small-space "sketch" Ax of the original dataset. Note that sampling can fall under the umbrella of linear sketching in the case where the random matrix only contains a single one as the nonzero entry in each row. Unfortunately, linear sketching again cannot handle the implicit deletions of the sliding window model, since it is not entirely clear how to "undo" the effect of each expired element in the linear sketch Ax.

Adapting insertion-only streaming algorithms to the sliding window model. Algorithms for the sliding window model are often adapted from the insertion-only streaming model through either the exponential histogram framework [DGIM02] or its generalization, the smooth histogram framework [BO07]. These frameworks transform streaming algorithms for either an additive function (in the case of exponential histograms) or a smooth function (in the case of smooth histograms) into sliding window algorithms by maintaining a logarithmic number of instances of the streaming algorithm, starting at various timestamps during the stream. Informally, a function is smooth if once a suffix of a data stream becomes a $(1+\beta)$-approximation of the entire data stream for the function, then the suffix is always a $(1+\alpha)$-approximation, regardless of the subsequent updates in the stream. Thus at the end of the stream of say length m, two of the timestamps must "sandwich" the beginning of the window, i.e., there exists timestamps t_1 and t_2 such that $t_1 \leq m - W + 1 < t_2$. The main point of the smooth histogram is that the streaming algorithm starting at time t_1 must output a value that is a good approximation of the function on the sliding window due to the smoothness of the function. Therefore, the smooth histogram is a cornerstone of algorithmic design in the sliding window model and handles many interesting functions, such as L_p norm estimation (and in particular the sum), longest increasing subsequence, geometric mean, distinct elements estimation, and counting the frequency of a specific item.

On the other hand, there remain interesting functions that are not smooth, such as clustering [BLLM16, BEL+20, EMMZ21], submodular optimization [CNZ16, ELVZ17], sampling [JWZ21], regression and low-rank approximation [BDM+20, UU21], and crucially for our purposes, heavy hitters [BGO14, BGL+18, Upa19, WZ21]. These problems cannot be handled by the smooth histogram framework and thus for these problems, sliding windows algorithms were developed utilizing the specific properties of the objective functions.

Previous Work in the DP setting. Among the previous literature, the work most related to the subject of our study is [Upa19] who proposed the study of differentially private L_1-heavy hitter algorithms in the sliding window. Although [Upa19] gave a continual release algorithm, which was later improved by [HQYC21], the central focus of our work is the "one-shot" setting, where the algorithm releases a single set of statistics at the end of the stream, because permitting a single interaction with the data structure can often achieve better guarantees for both the space complexity and the utility of the algorithm. Indeed, in this paper we present L_2-heavy hitter algorithms for both the continual release and the one-shot settings, but the space/accuracy tradeoffs in the latter are much better than the former. [Upa19] also proposed a "one-shot" algorithm, which empirically performs well, but lacks the theoretical guarantees claimed in the paper. We refer to Section 1.3 for more details.

Privately releasing heavy-hitters in other big data models has also received significant attention. [DNP+10] introduced the problem of L_1-heavy hitters and other problems in the *pan-privacy* streaming model, where the goal is to preserves differential privacy even if the internal memory

of the algorithm is compromised, while [CLSX12] considered the problem of continually releasing L_1-heavy hitters in a stream. The heavy-hitter problem has also been extensively studied in the local model [BS15, DKY17, AS19, BNS19, BNST20]. In the local model, individual users locally add privacy to their data, e.g., through randomized response, before sending their private information to a central and possibly untrusted server to aggregate the statistics across all users.

1.3 Overview of Our Techniques

We first use the smooth histogram to obtain a constant factor approximation to the L_2 norm of the sliding window similar to existing heavy-hitter non-DP algorithms in the sliding window model [BGO14, BGL+18]. We maintain a series of timestamps $t_1 < t_2 < \ldots < t_s$ for $s = \mathcal{O}(\log n)$, such that $L_2(t_1 : m) > L_2(t_2 : m) > \ldots > L_2(t_s : m)$ and $t_1 \leq m - W + 1 < t_2$. Hence, $L_2(t_1 : m)$ is a constant factor approximation to $L_2(m - W + 1 : m)$, which is the L_2 norm of the sliding window. For each timestamp t_i with $i \in [s]$, we also run an L_2-heavy hitter algorithm $\textsc{CountSketch}_i$, which outputs a list $\mathcal{L}_i$ of size at most $\mathcal{O}\left(\frac{1}{\alpha^2}\right)$ that contains the L_2-heavy hitters of the suffix of the stream starting at time t_i, as well as approximations to each of their frequencies. It might be tempting to simply output a noisy version of the list $\mathcal{L}_1$ output by $\textsc{CountSketch}_1$, since t_1 and t_2 sandwich the start of the sliding window, $m - W + 1$. Indeed, this is the approach by [Upa19], although they only consider the L_1-heavy hitter algorithm CountMin because they study the weaker L_1-heavy hitter problem and they do not need to run a norm estimation algorithm because L_1 can be computed exactly. However, [BGO14, BGL+18] crucially note that $\mathcal{L}_1$ can also include a number of items that are heavy-hitters with respect to the suffix of the stream starting at time t_1 but *are not* heavy-hitters in the sliding window because many or even all of them appeared before time $m - W + 1$. Thus although $\mathcal{L}_1$ can guarantee that all the L_2-heavy hitters are reported by considering a lower threshold, say $\frac{\alpha}{2}$, the frequencies of each reported heavy-hitter can be arbitrarily inaccurate.

Observe it does not suffice to instead report the L_2-heavy hitters starting from time t_2. Although this will remove the false-positive issue of outputting items that are not heavy-hitters, there is now a false-negative issue; there may be heavy-hitters that appear after time $m - W + 1$ but before time t_2 that will not be detected by $\textsc{CountSketch}_2$. Hence, there may be heavy-hitters of the sliding window that are not reported by $\mathcal{L}_2$. See Figure 1 for an example.

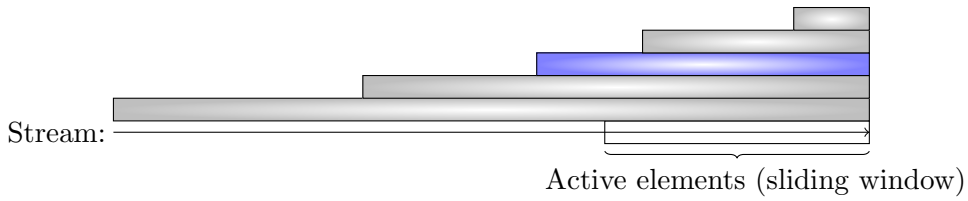

Fig. 1: Informally, we start a logarithmic number of streaming algorithms (the grey rectangles) at different points in time. We call the algorithm with the shortest substream that contains the active elements at the end of the stream (the blue rectangle). The challenge is that there may be heavy-hitters with respect to the blue rectangle that only appear before the active elements and therefore may be detected as heavy-hitters of the sliding window even though they are not.

Approximate counters. The fix by [BGO14, BGL+18] that is missed by [Upa19] is to run approximate counters for each item $k \in [n]$ reported by some heavy-hitter algorithm $\textsc{CountSketch}_i$, i.e., there exists $i \in [s]$ such that $k \in \mathcal{L}_i$. An approximate counter is simply a sliding window algorithm that reports a constant factor approximation to the frequency of a specific item $k \in [n]$. One way to achieve an approximate counter is to use the smooth histogram framework [BO07], but we show that an improved accuracy can be guaranteed if the maintenance procedure instead considers additive error rather than multiplicative error. Given the approximate counter that reports an estimate $\widehat{f}_k$ as the frequency for an item $k \in [n]$, we can then compare $\widehat{f}_k$ to the estimated L_2 norm of the sliding window to determine whether k could possibly be an L_2-heavy hitter. This rules out the false positives that can be returned in $\mathcal{L}_1$ without incurring false negatives omitted by $\mathcal{L}_2$.

Large sensitivity of subroutines. So far we have only discussed the techniques required to release L_2-heavy hitters in the non-DP setting. In order to achieve differential privacy, a first attempt might be to add Laplacian noise to each of the procedures. Namely, we would like to add Laplacian noise to the estimate of the L_2 norm of the sliding window and the frequency of each reported heavy-hitter. However, since both the estimate of the L_2 norm of the sliding window and the frequency of each reported heavy-hitter is governed by the timestamps $t_1, \ldots, t_s$, then the sensitivity of each quantity can be rather large. In fact, if the frequency of each reported heavy-hitter has sensitivity $\alpha \cdot L_2(m - W + 1 : m)$ through the approximate counters, then with high probability, the Laplacian noise added to the frequency of some reported heavy-hitter will completely dominate the actual frequency of the item to the point where it is no longer possible to identify the heavy-hitters. Thus the approximate counters missed by [Upa19] actually pose a significant barrier to the privacy analysis of the algorithm.

Noisy timestamps. Instead of adding large Laplacian noise to each of the estimates, another possible attempt might be to make the timestamps in the histogram themselves noisy, e.g., by adding Laplacian noise to each of the timestamps. At first, it seems that the timestamps crucially govern the approximation guarantees by the smooth histogram and so adding noise would disrupt any sort of quality-of-approximation guarantee. However, upon closer examination, one can observe that due to the properties of L_2 and the count of an item, the Laplacian noise added to a timestamp would induce only a small additive error on each of the estimations. Unfortunately, we would no longer have sketches that correspond to the noisy timestamps. That is, suppose the smooth histogram maintains a heavy-hitter algorithm $\textsc{CountSketch}_1$ starting at a time t_1. Prior to releasing the statistics, suppose we add noise to the value of t_1 and obtain a noisy timestamp $\tilde{t_1}$. We would like to release the statistics of the dataset that begins with the $\tilde{t_1}$-th update of the stream, but it is not clear how to do so because we do not actually have a streaming algorithm starting at a time $\tilde{t_1}$. We could use $\textsc{CountSketch}_1$ as a proxy but that defeats the purpose of adding noise to the timestamp in the first place.

Lower smooth sensitivity through better approximations. Instead, we guarantee differential privacy using the notion of smooth sensitivity [NRS07]. The idea is the following — given an α-approximation algorithm $\mathcal{A}$ for a function with sensitivity Δ_f, we would like to intuitively say the approximation algorithm has sensitivity $\alpha\Delta_f$. Unfortunately, this is not true because $\mathcal{A}(X)$ may report $\alpha \cdot f(X)$ and $\mathcal{A}(Y)$ may report $\frac{1}{\alpha} \cdot f(Y)$ for adjacent datasets X and Y. However, if $\mathcal{A}$ is instead a $(1+\alpha)$-approximation algorithm, then difference of the output of $\mathcal{A}$ on X and Y

can be bounded by $\alpha \cdot f(X) + \alpha \cdot f(Y) + \Delta_f$ through a simple triangle inequality, *conditioned on the correctness* of $\mathcal{A}$. In other words, if α is sufficiently small, then we can show that the *local sensitivity* of $\mathcal{A}$ is sufficiently small, which allows us to control the amount of Laplacian noise that must be added through existing mechanisms for smooth sensitivity. Unfortunately, if $\mathcal{A}$ is not correct, then even the local sensitivity could be quite large; we handle these cases separately by analyzing the smooth sensitivity of an approximation algorithm that is always correct and then arguing indistinguishability through statistical distance. Therefore, we can set the accuracy of the L_2 norm estimation algorithm, each L_2-heavy hitter algorithm, and each approximate counter algorithm to be sufficiently small and finally we can add Laplacian noise to each procedure without significantly impacting the final check of whether the estimated frequency for each item exceeds the heavy-hitter threshold.

Pure differential privacy for L_1-heavy hitters in the sliding window model. Due to the linearity of L_1, our algorithm for differentially private L_1-heavy hitters in the sliding window model is significantly simpler than the L_2-heavy hitters algorithm. For starters, each set of c updates must contribute exactly c to the L_1 norm, whereas their contribution to the L_2 norm depends on the particular coordinates they update. Therefore, not only do we not require an algorithm to approximate the L_1 norm of the active elements of the sliding window, but also we can fix a set of static timestamps in the smooth histogram, so we do not need to perform the same analysis to circumvent the sensitivity of the timestamps. Instead, it suffices to initialize a *deterministic* L_1-heavy hitter algorithm at each timestamp and maintain deterministic counters for each reported heavy-hitter. Pure differential privacy then follows from the lack of failure conditions in the subroutines, which was not possible for L_2-heavy hitters.

2 Preliminaries

For an integer $n > 0$, we use the notation $[n] := \{1, \ldots, n\}$. We use the notation $\text{poly}(n)$ to represent a constant degree polynomial in n and we say an event occurs *with high probability* if the event holds with probability $1 - \frac{1}{\text{poly}(n)}$. We say that $\mathcal{A}$ is an (α, δ)-approximation algorithm for the function $f : \mathcal{U}^* \to \mathbb{R}$ if for any $X \in \mathcal{U}^*$, we have that

$$\mathbf{Pr}\left[(1-\alpha)f(X) \leq \mathcal{A}(X) \leq (1+\alpha)f(X)\right] \geq 1 - \delta.$$

2.1 Differential Privacy

In this section, we first introduce simple or well-known results from differential privacy. We say that streams $\mathfrak{S}$ and $\mathfrak{S}'$ are *neighboring*, if there exists a single update $i \in [m]$ such that $u_i \neq u'_i$, where $u_1, \ldots, u_m$ are the updates of $\mathfrak{S}$ and $u'_1, \ldots, u'_m$ are the updates of $\mathfrak{S}'$.

Definition 2.1 (L_1 sensitivity)**.** *The L_1 sensitivity of a function $f : \mathcal{U}^* \to \mathbb{R}^k$ is defined by*

$$\Delta_f = \max_{x,y \in \mathcal{U}^*|, \|x-y\|_1 = 1} \|f(x) - f(y)\|_1.$$

The L_1 sensitivity of a function f bounds the amount that f can change when a single coordinate of the input to f changes and is often used to parameterize the amount of added noise to ensure differential privacy. For example, random noise may be generated from the Laplacian distribution:

Definition 2.2 (Laplace distribution)**.** *We say a random variable X is drawn from a Laplace distribution with mean μ and scale $b > 0$ if the probability density function of X at x is $\frac{1}{2b}\exp\left(-\frac{|x-\mu|}{b}\right)$. We use the notation $X \sim \mathsf{Lap}(b)$ to denote that X is drawn from the Laplace distribution with scale b and mean $\mu = 0$.*

Fact 2.3. *If $Y \sim \mathsf{Lap}(b)$, then $\Pr[|Y| \geq \ell \cdot b] = \exp(-\ell)$.*

In particular, the Laplace mechanism adds Laplacian noise with scale Δ_f, the L_1 sensitivity of the function f.

Definition 2.4 (Laplace mechanism)**.** *Given a function $f : \mathcal{U}^* \to \mathbb{R}^k$, the Laplace mechanism is defined by:*

$$\mathcal{M}_L(x, f, \varepsilon) = f(x) + (X_1, \ldots, X_k),$$

where $X_i \sim \mathsf{Lap}(\Delta_f/\varepsilon)$ for $1 \leq i \leq k$.

The Laplace mechanism is one of the most common methods of guaranteeing pure differential privacy.

Theorem 2.5 ([DR14])**.** *The Laplace mechanism preserves $(\varepsilon, 0)$-differential privacy when Δ_f is the L_1 sensitivity.*

We define the following notion of local L_1 sensitivity for a fixed input, which can be much smaller than the (global) L_1 sensitivity.

Definition 2.6 (Local sensitivity)**.** *For $f : \mathcal{U}^* \to \mathbb{R}$ and $x \in \mathcal{U}^*$, the local sensitivity of f at x is defined as*

$$LS_f(x) = \max_{y : \|x-y\|_1 = 1} \|f(x) - f(y)\|_1.$$

Unfortunately, the local sensitivity can behave wildly for specific algorithms. Thus we have the following definition that smooths such behavior for local sensitivity.

Definition 2.7 (Smooth upper bound on local sensitivity)**.** *For $\beta > 0$, a function $S : \mathcal{U}^* \to \mathbb{R}$ is a β-smooth upper bound on the local sensitivity of $f : \mathcal{U}^* \to \mathbb{R}$ if*

(1) For all $x \in \mathcal{U}^*$, we have $S(x) \geq LS_f(x)$.

(2) For all $x, y \in \mathcal{U}^*$ with $\|x - y\|_1 = 1$, we have $S(x) \leq e^\beta \cdot S(y)$.

Even though the local sensitivity can be much smaller than the global L_1 sensitivity, the Laplace mechanism as defined in Definition 2.4 adds noise scaling with the global L_1 sensitivity. Hence it seems natural to hope for a mechanism that adds less noise. The following result shows that this is indeed possible.

Theorem 2.8 (Corollary 2.4 in [NRS07])**.** *Let $f : \mathcal{U}^* \to \mathbb{R}$ and $S : \mathcal{U}^* \to \mathbb{R}$ be a β-smooth upper bound on the local sensitivity of f. If $\beta \leq \frac{\varepsilon}{2\ln(2/\delta)}$ and $\delta \in (0, 1)$, then the mechanism that outputs $f(x) + X$, where $X \sim \mathsf{Lap}\left(\frac{2S(x)}{\varepsilon}\right)$ is (ε, δ')-differentially private, for $\delta' = \frac{\delta}{2}\left(1 + \exp\left(\frac{\varepsilon}{2}\right)\right)$.*

We have the following theorems on the composition and post-processing of differentially private mechanisms.

Theorem 2.9 (Composition and post-processing of differential privacy [DR14]). *Let $\mathcal{M}_i : \mathcal{U}_i^* \to X_i$ be an $(\varepsilon_i, \delta_i)$-differential private algorithm for $i \in [k]$. Then $\mathcal{M}_{[k]}(x) = (\mathcal{M}_1(x), \ldots, \mathcal{M}_k(x))$ is $\left(\sum_{i=1}^k \varepsilon_i, \sum_{i=1}^k \delta_i\right)$-differentially private. Moreover, if $g_i : X_i \to X_i'$ is an arbitrary random mapping, then $g_i(\mathcal{M}_i(x))$ is $(\varepsilon_i, \delta_i)$-differntially private.*

Theorem 2.10 (Advanced composition of differential privacy [DR14]). *For all $\varepsilon, \delta \geq 0$ and $\delta' > 0$, the advanced composition of k algorithms, each of which is (ε, δ)-differentially private, is $(\tilde{\varepsilon}, \tilde{\delta})$-differentially private, where*

$$\tilde{\varepsilon} = \varepsilon\sqrt{2k\ln(1/\delta')} + k\varepsilon\left(\frac{e^\varepsilon - 1}{e^\varepsilon + 1}\right), \qquad \tilde{\delta} = k\delta + \delta'.$$

2.2 Norm Estimation

In this section, we introduce preliminaries for norm or frequency moment estimation.

Definition 2.11 (Norm/moment estimation). *Given $p > 0$ and a frequency vector $f \in \mathbb{R}^n$, we define the moment of f by $F_p(f) = \sum_{k=0}^n f_k^p$ and the L_p norm of f by $\|f\|_p = L_p(f) = (F_p(f))^{1/p}$. For an accuracy parameter $\alpha \in (0,1)$, the* F_p moment estimation problem *is to output an estimated moment $\widehat{F}$ such that $|\widehat{F} - F_p(f)| \leq \alpha F_p(f)$ and the* norm estimation problem *is to output an estimated norm $\widehat{L}$ such that $|\widehat{L} - L_p(f)| \leq \alpha L_p(f)$.*

We note that L_p is not a norm for $p \in (0,1)$, but the problem is nevertheless well-defined and also well-motivated due to the importance of frequency moment estimation for $p \in (0,1)$. Specifically, the norm and moment estimation problems are often used interchangeably because a $(1+\alpha)$-approximation to L_p also gives a $(1+\alpha)^p = (1+\mathcal{O}(\alpha))$-approximation to F_p, for sufficiently small α. Thus an algorithm that achieves $(1+\alpha)$-approximation to L_p given an accuracy parameter $\alpha > 0$ can also be adjusted to achieve a $(1+\alpha)$-approximation to F_p by scaling the input accuracy α.

Theorem 2.12 (Norm estimation algorithm AMS [AMS99, BCIW16, BDN17]). *Given an accuracy parameter $\alpha > 0$ and a failure probability $\delta \in (0,1)$, there exists a one-pass streaming algorithm* AMS *for the L_2 norm estimation problem, using $\mathcal{O}\left(\frac{\log n + \log m}{\alpha^2} \log \frac{\log m}{\alpha\delta}\right)$ bits of space and $\mathcal{O}\left(\frac{1}{\alpha^2} \log \frac{\log m}{\alpha\delta}\right)$ operations per time.*

We first recall the AMS algorithm for F_2 frequency estimation, which is formalized in Algorithm 1 for constant probability of success. The algorithm outputs a $(1+\alpha)$-approximation to the second moment of the frequency vector using $\mathcal{O}\left(\frac{1}{\alpha^2}\log n \log\frac{1}{\delta}\right)$ space. The algorithm first generates a random sign vector s of length n, so that $s_i \in \{-1, +1\}$ for each $i \in [n]$. The algorithm then computes the inner product $Z = \langle s, f\rangle$, so that Z^2 is an unbiased estimator of F_2 with variance $\mathcal{O}\left(F_2^2\right)$. It follows from a standard variance reduction argument through Chebyshev's inequality that the mean of $\mathcal{O}\left(\frac{1}{\alpha^2}\right)$ such inner products is a $(1 \pm \alpha)$ approximation of F_2 with constant probability and thus the median of $\mathcal{O}\left(\log\frac{1}{\delta}\right)$ such means is a $(1 \pm \alpha)$ approximation of F_2 with probability at least $1 - \delta$. Thus by taking the median of $\mathcal{O}\left(\log\frac{1}{\delta}\right)$ independent instances of Algorithm 1 boosts the probability of success to $1 - \delta$. Similarly, the square root of the median is a $(1 \pm \alpha)$ approximation of L_2 with probability at least $1 - \delta$, though one could also view the L_2 estimation algorithm as taking the median of means of the absolute value of each inner product Z, which is equivalent to taking the square root of Z^2.

We first show the global L_1 sensitivity of the L_2 norm.

Algorithm 1 Algorithm AMS for F_2 estimation

Input: Stream $\mathfrak{S}$, threshold/accuracy parameter $\alpha \in (0,1)$
Output: $(1+\alpha)$-approximation of F_2 with probability at least $2/3$

Set $r = \mathcal{O}\left(\frac{1}{\alpha^2}\right)$
Generate r random sign vectors $s^{(1)}, \ldots, s^{(r)}$
Initialize sums $S_1, \ldots, S_r = 0$
for each update $u_i \in [n]$, $i \in [m]$ **do**
    for each $j \in [r]$ **do**
        Update $S_j \leftarrow S_j + s_{u_i}^{(j)}$
return $\frac{1}{r}\sum_{j=1}^{r}(S_j)^2$

Lemma 2.13 (Sensitivity of L_2)**.** *Let f and f' be frequency vectors on a universe of size n such that $n-2$ coordinates of f and f' have the same value and the remaining two coordinates differ by exactly one. Then $|L_2(f) - L_2(f')| \leq 2$.*

Proof. Observe that by the concavity of the square root function, we have $\sqrt{(x+1)^2+y} - \sqrt{x^2+y} \leq 1$ for all $x, y \geq 0$. Let i and j be the indices where the coordinates of f and f' differ, i.e., $|f[i] - f'[i]| = 1$, $|f[j] - f'[j]| = 1$, and $f[k] = f'[k]$ for all $k \in [n] \setminus \{i, j\}$. Now we define f'' to be an intermediate frequency vector such that it differs by only one coordinate to f and f', respectively, i.e., $f''[i] = f[i]$, $f''[j] = f'[j]$, and $f''[k] = f[k] = f'[k]$ for all $k \in [n] \setminus \{i, j\}$. Then by the previous observation and the triangle inequality, we have $|L_2(f) - L_2(f')| \leq |L_2(f) - L_2(f'')| + |L_2(f'') - L_2(f')| \leq 1 + 1 = 2$. □

2.3 Heavy Hitters

In this section, we formally introduce the L_p-heavy hitter problem and the algorithm CountSketch, which is commonly used to find the L_2-heavy hitters.

Definition 2.14 (L_p-heavy hitter problem)**.** *Given an accuracy/threshold parameter $\alpha \in (0,1)$, $p > 0$, and a frequency vector $f \in \mathbb{R}^n$, report all coordinates $k \in [n]$ such that $f_k \geq \alpha\, L_p(f)$ and no coordinates $j \in [n]$ such that $f_j \leq \frac{\alpha}{2}\, L_p(f)$. For each reported coordinate $k \in [n]$, also report an estimated frequency $\widehat{f_k}$ such that $|\widehat{f_k} - f_k| \leq \frac{\alpha}{4}\, L_p(f)$.*

Theorem 2.15 (Heavy-hitter algorithm CountSketch [CCF04])**.** *Given an accuracy parameter $\alpha > 0$ and a failure probability $\delta \in (0,1)$, there exists a one-pass streaming algorithm* CountSketch *for the L_2-heavy hitter problem that uses $\mathcal{O}\left(\frac{1}{\alpha^2}\log\frac{n}{\delta}\right)$ words of space and $\mathcal{O}\left(\log\frac{n}{\delta}\right)$ update time.*

The CountSketch data structure [CCF04] is an r by b table, which can be thought of as r rows of b buckets, each with counters that are initialized to zero. In each row $j \in [r]$, each item of the universe $i \in [n]$ is assigned to one of the b buckets by a hash function $h^{(j)} : [n] \to [b]$, so that the bucket for item i is $h^{(j)}(i)$. If i appears in the stream, then the random sign $s^{(j)}(i)$ is added to the counter corresponding to the bucket assigned to the item in row j for each $j \in [r]$. At the end of the stream, the mean across all r rows of the magnitude of the counters for the buckets assigned to i corresponds to the estimate of the frequency of i. Due to the $\mathcal{O}(\log n)$ rows, the algorithm has failure probability $1 - \frac{1}{\text{poly}(n)}$. For completeness, we provide the full details in Algorithm 2.

Algorithm 2 Algorithm CountSketch for heavy-hitter estimation

Input: Stream $\mathfrak{S}$, threshold/accuracy parameter $\alpha \in (0,1)$
Output: L_2-Heavy hitter algorithm

Set $r = \mathcal{O}(\log n)$, $b = \mathcal{O}\left(\frac{1}{\alpha^2}\right)$
Generate r hash functions $h^{(1)}, \ldots, h^{(r)} : [n] \to [b]$ and $s^{(1)}, \ldots, s^{(r)} : [n] \to \{-1, +1\}$
Initialize sums $S_{i,j} = 0$ for $(i,j) \in [r] \times [b]$
for each update $u_i \in [n]$, $i \in [m]$ **do**
    for each $j \in [r]$ **do**
        Set $b_{i,j} = h^{(j)}(u_i)$ and $s_{i,j} = s^{(j)}(u_i)$
        Update $S_{j,b_{i,j}} = S_{j,b_{i,j}} + s_{i,j}$
for each $i \in [n]$ **do**
    Set $b_{i,j} = h^{(j)}(u_i)$ for each $j \in [r]$
    return $\widehat{f_i} = \frac{1}{r} \sum_{j \in [r]} |S_{j,b_{i,j}}|$ as the estimated frequency for f_i

2.4 Sliding Window Model

In this section, we introduce simple or well-known results for the sliding window model.

Definition 2.16 (Sliding window model)**.** *Given a universe $\mathcal{U}$ of items, which we associate with $[n]$, let a stream $\mathfrak{S}$ of length m consist of updates $u_1, \ldots, u_m$ to the universe $\mathcal{U}$, so that $u_i \in [n]$ for each $i \in [m]$. After the stream, a window parameter W is given, which induces the frequency vector $f \in \mathbb{R}^n$ so that $f_k = |\{i : u_i = k \wedge i \geq m - W + 1\}|$ for each $k \in [n]$. In other words, each coordinate k of the frequency vector is the number of updates to k within the last W updates.*

We say A and B are *adjacent* substreams of a stream $\mathfrak{S}$ of length m if A consists of the updates $u_i, \ldots, u_j$ and B consists of the updates $u_{j+1}, \ldots, u_k$ for some $i, j, k \in [m]$. We have the following definition of a smooth function for the purposes of sliding window algorithms, not to be confused with the smooth sensitivity definition for differential privacy.

Definition 2.17 (Smooth function)**.** *Given adjacent substreams A and B, a function $g : \mathcal{U}^* \to \mathbb{R}$ is (α, β)-smooth if $(1-\beta)g(A \cup B) \leq g(B)$ implies $(1-\alpha)g(A \cup B \cup C) \leq g(B \cup C)$ for some parameters $0 < \beta \leq \alpha < 1$ and* any *adjacent substream C.*

Smooth functions are a key building block in the smooth histogram framework by [BO10], which creates a sliding window algorithm for a large number of functions using multiple instances of streaming algorithms starting at different points in time. See Algorithm 3 for more details on the smooth histogram.

Theorem 2.18 (Smooth histogram [BO10])**.** *Given accuracy parameter $\alpha \in (0,1)$, failure probability $\delta \in (0,1)$ and an (α, β)-smooth function $g : \mathcal{U}^m \to \mathbb{R}$, suppose there exists an insertion-only streaming algorithm $\mathcal{A}$ that outputs a $(1+\alpha)$-approximation to g with high probability using space $\mathcal{S}(\alpha, \delta, m, n)$ and update time $\mathcal{T}(\alpha, \delta, m, n)$. Then there exists a sliding window algorithm that outputs a $(1+\alpha)$-approximation to g with high probability using space $\mathcal{O}\left(\frac{1}{\beta}(\mathcal{S}(\beta, \delta, m, n) + \log m) \log m\right)$ and update time $\mathcal{O}\left(\frac{1}{\beta}(\mathcal{T}(\beta, \delta, m, n)) \log m\right)$.*

The following smoothness parameters for the F_p frequency moment and L_p norm functions suffice for our purposes:

Algorithm 3 Smooth histogram [BO10]

Input: Stream $\mathfrak{S}$, accuracy parameter $\rho \in (0,1)$, streaming algorithm $\mathcal{A}$ for $(\rho, \beta(\rho))$-smooth function

Output: $(1+\rho)$-approximation of predetermined function with probability at least $1-\delta$

$H \leftarrow \emptyset$
for each update u_t with $t \in [m]$ **do**
    $H \leftarrow H \cup \{t\}$
    for each time $t_s \in H$ **do**
        Let x_s be the output of $\mathcal{A}$ with failure probability $\frac{\delta}{\text{poly}(n,m)}$ starting at time t_s and ending at time t.
        if $x_{s-1} \leq \left(1 - \frac{\beta(\rho)}{2}\right) x_{s+1}$ **then**
            Delete t_s from H and reorder the indices in H
Let s be the smallest index such that $t_s \in H$ and $t_s \leq m - W + 1$.
Let x_s be the output of $\mathcal{A}$ starting at time t_s at time t.
return x_s

Lemma 2.19 ([BO10]). *For any $\rho \in (0,1)$, the F_p and L_p functions are $\left(\rho, \frac{\rho^p}{p}\right)$-smooth for $p \geq 1$ and (ρ, ρ)-smooth for $0 < p \leq 1$.*

Approximate frequency in the sliding window model. Finally, we present a deterministic algorithm Counter that can be parametrized to give an additive M-approximation to the estimated frequency $\widehat{f_i}$ of a particular element $i \in [n]$ in the sliding window model. The algorithm initializes counters for i starting at multiple times in the stream. To maintain additive error M, the algorithm removes a counter starting at a certain time, if the difference between the previous counter and the next counter is at most M, since it suffices to simply use the previous counter instead. For more details, see Algorithm 4.

Algorithm 4 Algorithm Counter for frequency estimation in the sliding window model

Input: Stream $\mathfrak{S}$, window parameter $W > 0$, additive error M, index $i \in [n]$

Output: $\widehat{f_i}$ such that $0 \leq f_i - \widehat{f_i} \leq M$

$H \leftarrow \emptyset$
for each update $u_t \in [n]$ with $t \in [m]$ **do**
    if $u_t = i$ **then**
        $H \leftarrow H \cup \{t\}$
    for each time $t_s \in H$ **do**
        Let c_s be the number of instances of i from t_s.
        if $c_{s-1} - c_{s+1} < M$ **then**
            Delete t_s from H and re-index H
Let s be the largest index such that $t_s \in H$ and $t_s > m - W + 1$.
Let $\widehat{f_i}$ be the number of instances of i from t_s.
return $\widehat{f_i}$

Lemma 2.20. *There exists a deterministic algorithm* Counter *that outputs an additive M approximation to the frequency of an element $i \in [n]$ in the sliding window model. The algorithm uses $\mathcal{O}\left(\frac{f_i}{M} \log m\right)$ bits of space.*

3 Differentially Private Heavy-Hitters in the Sliding Window Model

In this section, we give a private algorithm for L_2-heavy hitters in the sliding window model. Our algorithm will initially use a smooth histogram approach by instantiating a number of L_2 norm estimation algorithm starting at various timestamps in the stream. Through a sandwiching argument, these L_2 norm estimation algorithms will provide a constant factor approximation to the L_2 norm of the sliding window, which will ultimately allow us to determine whether elements of the stream are heavy-hitters. Moreover, by using a somewhat standard smooth sensitivity argument, we can show that these subroutines can be maintained in a way that preserves differential privacy.

To identify a subset of elements that can be heavy-hitters, we also run a private L_2-heavy hitters algorithm starting at each timestamp. Unfortunately, because the timestamps do not necessarily coincide with the beginning of the sliding window, it may be possible that depending on our approach, we may either output a number of elements with very low, possibly even zero, frequency, or we may neglect to output a number of heavy-hitters. To overcome this issue, we maintain private approximate counters for each item that is reported by our private L_2-heavy hitters algorithms, which allows us to rule out initially reported false positives without incurring false negatives.

The crucial observation is all elements that are heavy-hitters with respect to the sliding window must first be reported by our heavy-hitter algorithm at some timestamp (not necessarily corresponding to the beginning of the sliding window). Although our private approximate counter for a universe element i is initiated only after the heavy-hitter algorithm outputs element i, the approximate counter only misses a small number of the instances of i, because we run our heavy-hitter algorithm with a significantly lower threshold. Hence the approximate counter subroutine will still estimate the frequencies of the potential heavy-hitters elements with a sufficiently small error that is enough to differentiate whether an element is heavy with respect to the sliding window. We give the algorithm in full in Algorithm 5.

We first describe the procedure for the approximate frequency estimation for each reported heavy-hitter. Let $\textsc{CountSketch}_a$ be an L_2-heavy hitter algorithm starting at timestamp t_a, where $a = \max\{i \in [s] : t_i \leq m - W + 1\}$ on window query $W > 0$. For a coordinate $k \in [n]$ that is reported by $\textsc{CountSketch}_a$ from times t through m, we use Counter to maintain a number of timestamps such that the frequency of k on the suffixes induced by the timestamps are arithmetically increasing by roughly $\alpha^2 L_2(f)/16$. We emphasize that we run the Counter for each reported heavy-hitter in the same pass as the rest of the algorithm.

Lemma 3.1. *Let $\mathcal{E}$ be the event that (1) the smooth histogram data structure does not fail, (2) all instances of* CountSketch *do not fail, and (3) $X \leq \frac{L_2(f)}{10}$ and $\max_{j\in[n]}(Y_j, Z_j) \leq \frac{\alpha L_2(f)}{10}$. Let $\textsc{CountSketch}_a$ be the instance of* CountSketch *starting at time t_a. Conditioned on $\mathcal{E}$, then for each reported heavy-hitter k by $\textsc{CountSketch}_a$, Algorithm 5 outputs an estimated frequency $\widehat{f_k}$ such that*

$$|f_k - \widehat{f_k}| \leq \frac{\alpha^3 \varepsilon}{500 \log n} L_2(f).$$

Algorithm 5 Differentially private sliding window algorithm for L_2-heavy hitters

Input: Stream $\mathfrak{S}$, accuracy parameter $\alpha \in (0,1)$, differential privacy parameters $\varepsilon, \delta > 0$, window parameter $W > 0$, size n of the underlying universe, upper bound m on the stream length

Output: A list $\mathcal{L}$ of L_2-heavy hitters with approximate frequencies

1: Process the stream $\mathfrak{S}$, maintaining timestamps $t_1, \ldots, t_s$ at each time $t \in [m]$ so that for each $i \in [s]$, either $i = s$, $t_{i+1} = t_i + 1$ or $L_2(t_i, t) \le \left(1 + \left(\frac{\varepsilon}{1000 \log m}\right)^2\right) L_2(t_{i+1}, t)$ through a smooth histogram with failure probability $\frac{\delta}{2m^2}$

2: Implement heavy-hitter algorithm CountSketch on the substream starting at t_i for each $i \in [s]$ with threshold $\frac{\alpha^3 \varepsilon}{500 \log m}$ and failure probability $\frac{\delta}{2m^2}$

3: Set $a = \max\{i \in [s] : t_i \le m - W + 1\}$ on window query $W > 0$

4: Set $\widehat{L_2}$ to be an $\left(1 + \frac{\varepsilon}{500 \log m}\right)$-approximation to $L_2(t_a, t)$ from the smooth histogram and $X \leftarrow \mathsf{Lap}\left(\frac{1}{40 \log m} \widehat{L_2}\right)$

5: **for** each heavy-hitter $k \in [n]$ reported by CountSketch starting at t_a **do**

6: Run Counter with additive error $\frac{\alpha^3 \varepsilon}{1000 \log m} \widehat{L_2}$ for each reported heavy-hitter

7: Let $\widehat{f_k}$ be the approximate frequency reported by Counter

8: $Y_k \leftarrow \mathsf{Lap}\left(\frac{\alpha}{75 \log m} \widehat{L_2}\right)$, $Z_k \leftarrow \mathsf{Lap}\left(\frac{\alpha}{75 \log m} \widehat{L_2}\right)$, $\widetilde{f_k} = \widehat{f_k} + Z_k$

9: **if** $\widetilde{f_k} \ge \frac{3\alpha}{4}\left(\widehat{L_2} + X\right) + Y_k$ **then**

10: $\mathcal{L} \leftarrow \mathcal{L} \cup \{(k, \widetilde{f_k})\}$

11: **return** $\mathcal{L}$

The algorithm uses $\mathcal{O}\left(\frac{1}{\alpha^6 \varepsilon^2} \log^3 m\right)$ *space and* $\mathcal{O}\left(\frac{\log^2 m}{\alpha^4 \varepsilon^2}\right)$ *update time per instance of* CountSketch.

Proof. An approximate frequency counter for k maintains a series of additional timestamps $t_1^{(k)} < \ldots < t_r^{(k)}$ that denotes suffixes of the stream, with the property that for each $i \in [r-2]$, the frequency of k from time $t_i^{(k)}$ to m is more than an additive $\frac{\alpha^3 \varepsilon}{500 \log m} \widehat{L_2}$ amount than the frequency of k from time $t_{i+2}^{(k)}$. Suppose without loss of generality that the stream has length m and coordinate $k \in [n]$ is reported by CountSketch_a from times t through m so that $t_1^{(k)} = t$. By a combination of casework on the time t compared to $m - W + 1$ and a standard sandwiching argument, we bound the error of the estimated frequency $\widehat{f_k}$ for k.

We have two cases: either $t < m - W + 1$ or $t \ge m - W + 1$. First suppose that $t < m - W + 1$ so that $t_1^{(k)} = t < m - W + 1$. Then there exists an index $i \in [r-1]$ such that $t_i^{(k)} \le m - W + 1 < t_{i+1}^{(k)}$. Let T_i be frequency of k between times $t_i^{(k)}$ to m and T_{i+1} be frequency of k between times $t_{i+1}^{(k)}$ to m, so that $T_{i+1} \le f_k \le T_i$. Then by Lemma 2.20 and the smoothness of L_1 of the frequency vector restricted to coordinate k in Lemma 2.19, we have $T_{i+1} \le T_i \le T_{i+1} + \frac{\alpha^3 \varepsilon}{1000 \log m} \widehat{L_2}$. Thus, both T_i and T_{i+1} are additive $\frac{\alpha^3 \varepsilon}{1000 \log m} \widehat{L_2}$-additive approximations to f_k, so that

$$|f_k - \widehat{f_k}| \le \frac{\alpha^3 \varepsilon}{1000 \log m} \widehat{L_2}.$$

Conditioning on $\mathcal{E}$, then by the smooth histogram and the smoothness of L_2, i.e., Lemma 2.19, we

have that $\widehat{L_2}(f) \le \left(1 + \frac{\varepsilon}{500 \log m}\right) L_2(f)$, so it follows that

$$|f_k - \widehat{f_k}| \le \frac{\alpha^3 \varepsilon}{500 \log m} L_2(f).$$

On the other hand, suppose $t \ge m - W + 1$. Then conditioning on $\mathcal{E}$ (the correctness of CountSketch$_a$), at most $\frac{\alpha^3 \varepsilon}{500 \log m} \widehat{L_2}$ instances of k have arrived between times t_a and t; otherwise item k would have also been reported as an L_2-heavy hitter by CountSketch$_a$ with threshold $\frac{\alpha^3 \varepsilon}{500 \log m}$ at time $t-1$. Since $t_1^{(k)} = t$, then the frequency T_i of k between times $t_i^{(k)}$ to m is exactly the number of updates to k since time t. Thus for $\widehat{f_k} = T_i$, we have that $|\widehat{f_k} - T_i| \le \frac{\alpha^3 \varepsilon}{500 \log m} L_2(f)$.

To analyze the space complexity, note that by the invariance of the timestamp maintenance, we have $T_i - T_{i+2} > \frac{\alpha^3 \varepsilon}{1000 \log m} \widehat{L_2}$ for each index $i \in [r]$ corresponding with timestamps $t_1^{(k)} < \ldots < t_r^{(k)}$. Similarly, the same invariance holds for the timestamps corresponding to the counters for other heavy-hitters $j \in [n]$ reported by CountSketch$_a$. Since $\widehat{L_2} > \left(1 + \frac{\varepsilon}{500 \log m}\right) L_2(f)$, then it follows that there can only be $\mathcal{O}\left(\frac{\log^2 m}{\alpha^3 \varepsilon^2}\right)$ timestamps. Each timestamp corresponds to a counter that is encoded using $\mathcal{O}(\log m)$ bits, so the counters use $\mathcal{O}\left(\frac{\log^3 m}{\alpha^3 \varepsilon^2}\right)$ space. On the other hand, by Theorem 2.15, each instance of CountSketch with threshold $\frac{\alpha^3 \varepsilon}{500 \log m}$ uses $\mathcal{O}\left(\frac{1}{\alpha^6 \varepsilon^2} \log^4 m\right)$ bits of space, which gives the space complexity.

To analyze the time complexity, note that each update corresponds to whether each of the counters corresponding to the $\mathcal{O}\left(\frac{\log^2 m}{\alpha^3 \varepsilon^2}\right)$ timestamps satisfies the invariant that $T_i - T_{i+2} > \frac{\alpha^3 \varepsilon}{1000 \log m} \widehat{L_2}$. Thus the update time is $\mathcal{O}\left(\frac{\log^2 m}{\alpha^3 \varepsilon^2}\right)$ operations. □

We first show that the list $\mathcal{L}$ output by Algorithm 5 does not contain any items with "low" frequency.

Lemma 3.2 (Low frequency items are not reported)**.** *Let $\mathcal{E}$ be the event that (1) the smooth histogram data structure does not fail, (2) all instances of* CountSketch *do not fail, and (3) $X \le \frac{L_2(f)}{10}$ and $\max_{j \in [n]}(Y_j, Z_j) \le \frac{\alpha L_2(f)}{10}$. Let f be the frequency vector induced by the sliding window parameter W and suppose $f_k \le \frac{\alpha}{2} L_2(f)$. Then conditioned on $\mathcal{E}$, $k \notin \mathcal{L}$.*

Proof. Let CountSketch$_a$ be the instance of CountSketch starting at time t_a. Let H_a be the set of heavy-hitters reported by CountSketch$_a$ with threshold $\alpha/16$ at time m. Then conditioned on $\mathcal{E}$, either (1) $k \notin H_a$ so that $k \notin \mathcal{L}$ and thus k will not be reported by Algorithm 5 or (2) $k \in H_a$, so that k is an $\alpha/16$ heavy-hitter of some suffix of the stream. In the latter case, an approximate frequency estimate $\widehat{f_k}$ for k is output by an instance of Counter.

By Lemma 3.1, we have that

$$|f_k - \widehat{f_k}| \le \frac{\alpha^3 \varepsilon}{500 \log m} L_2(f).$$

Thus, $\widehat{f_k} \le f_k + \alpha\, L_2(f)/8$. By the smooth histogram, we also have that $\left(1 + \frac{\varepsilon}{500 \log m}\right) \widehat{L_2} \ge L_2(f)$. Hence $f_k \le \frac{\alpha}{2} L_2(f)$ along with the assumption that $X \le \frac{L_2(f)}{10}$ and $\max_{j \in [n]}(Y_j, Z_j) \le \frac{\alpha L_2(f)}{10}$ conditioned on $\mathcal{E}$ implies that $\widehat{f_k} < \frac{3\alpha}{4}(\widehat{L_2} + X) + Y_k$ so k will not be added to $\mathcal{L}$ and thus not reported by Algorithm 5. □

We now show that the heavy-hitters are reported and bound the error in the estimated frequency for each reported item.

Lemma 3.3 (Heavy-hitters are estimated accurately)**.** *Let f be the frequency vector induced by the sliding window parameter W. Let $\mathcal{E}$ be the event that (1) the smooth histogram data structure does not fail, (2) all instances of* CountSketch *do not fail, and (3) $X \le \frac{L_2(f)}{10}$ and $\max_{j\in[n]}(Y_j, Z_j) \le \frac{\alpha L_2(f)}{10}$. Conditioned on $\mathcal{E}$, then $k \in \mathcal{L}$ for each $k \in [n]$ with $f_k \ge \alpha\, L_2(f)$. Moreover, for each item $k \in \mathcal{L}$,*

$$|f_k - \widehat{f_k}| \le \frac{\alpha^3\varepsilon}{500\log m}\, L_2(f).$$

Proof. Let $\textsc{CountSketch}_a$ be the instance of CountSketch starting at time t_a. Since $f_k \ge \alpha\, L_2(f)$ and $L_2(t_a, t) \le \left(1 + \frac{\varepsilon}{500\log m}\right) L_2(f)$ implies $f_k \ge \frac{\alpha}{16}\, L_2(t_a, t)$, then f_k will be reported by $\textsc{CountSketch}_a$, conditioned on $\mathcal{E}$ (the correctness of $\textsc{CountSketch}_a$). We thus have an approximate frequency estimate $\widehat{f_k}$ for k output by an instance of Counter. By Lemma 3.1,

$$|f_k - \widehat{f_k}| \le \frac{\alpha^3\varepsilon}{500\log m}\, L_2(f).$$

Furthermore, we have that $\widehat{L_2} \le \left(1 + \frac{\varepsilon}{500\log m}\right) L_2(t_a, t) \le \left(1 + \frac{\varepsilon}{500\log m}\right)^2 L_2(f)$. Given the assumption that $X \le \frac{L_2(f)}{10}$ and $\max_{j\in[n]}(Y_j, Z_j) \le \frac{\alpha L_2(f)}{10}$ conditioned on $\mathcal{E}$, then $\widehat{f_k} < \frac{3\alpha}{4}\,(\widehat{L_2} + X) + Y_k$ so $k \in \mathcal{L}$.

Let H_a be the set of heavy-hitters reported by $\textsc{CountSketch}_a$ with threshold $\frac{\alpha^3\varepsilon}{500\log m}$ at time m. For each $j \in \mathcal{L}$, we have that $j \in H_a$ so that an approximate frequency estimate $\widehat{f_j}$ for j is output by an instance of Counter. By Lemma 3.1, we have that $|f_j - \widehat{f_j}| \le \frac{\alpha^3\varepsilon}{500\log m}\, L_2(f)$, as desired. □

We now show that the event $\mathcal{E}$ conditioned by Lemma 3.1, Lemma 3.2, and Lemma 3.3 occurs with high probability.

Claim 3.4. *Let* $\mathbf{Pr}\left[\widehat{L_2}(f) \le 2L_2(f)\right] \le \delta$. *Suppose* $X \sim \mathsf{Lap}\left(\frac{1}{40\log m}\widehat{L_2}(f)\right)$, *and for* $j \in [n]$, $Y_j, Z_j \sim \mathsf{Lap}\left(\frac{\alpha}{75\log m}\widehat{L_2}(f)\right)$. *Then,* $\mathbf{Pr}\left[X \le \frac{L_2(f)}{10} \wedge \max_{j\in[n]}(Y_j, Z_j) \le \frac{\alpha L_2(f)}{10}\right] \ge 1 - \left(\frac{1}{m^2} + \frac{2}{m^{\frac{11}{4}}} + \delta\right)$.

Proof. Using Fact 2.3,

$$\mathbf{Pr}\left[|X| > \frac{\widehat{L_2}(f)}{20}\right] = \mathbf{Pr}\left[|X| > \frac{\widehat{L_2}(f)}{40\log m}\cdot 2\log m\right] = \frac{1}{m^2}$$

Also using Fact 2.3, for a fixed j,

$$\mathbf{Pr}\left[|Y_j| > \frac{\alpha\widehat{L_2}(f)}{20}\right] = \mathbf{Pr}\left[|Y_j| > \frac{\widehat{L_2}(f)}{75\log m}\cdot\frac{75}{20}\log m\right] = \frac{1}{m^{\frac{15}{4}}}$$

By a union bound over all $j \in [n]$,

$$\mathbf{Pr}\left[\max_{j\in[n]}|Y_j| > \frac{\alpha\widehat{L_2}(f)}{20}\right] \le \frac{n}{m^{\frac{15}{4}}}$$

Since Z_j is identically distributed and over $j \in [n]$, and $n < m$, we have that

$$\mathbf{Pr}\left[\max_{j\in[n]} |(Y_j, Z_j)| > \frac{\alpha\widehat{L_2}(f)}{20}\right] \leq \frac{2n}{m^{\frac{15}{4}}} \leq \frac{2}{m^{\frac{11}{4}}}$$

We know that with probability $1-\delta$, $\widehat{L_2}(f) \leq 2L_2(f)$, therefore, by a union bound, and using $n < m$,

$$\mathbf{Pr}\left[(\widehat{L_2}(f) > 2L_2(f)) \vee \left(\max_{j\in[n]} |(Y_j, Z_j)| > \frac{\alpha\widehat{L_2}(f)}{20}\right) \vee \left(|X| > \frac{1}{20}\widehat{L_2}(f)\right)\right] \leq \frac{2}{m^{\frac{11}{4}}} + \frac{1}{m^2} + \delta.$$

□

Lemma 3.5. *Let $\mathcal{E}$ be the event that (1) the smooth histogram data structure does not fail on either stream, (2) all instances of* CountSketch *do not fail, and (3) $X \leq \frac{L_2(f)}{10}$ and $\max_{j\in[n]}(Y_j, Z_j) \leq \frac{\alpha L_2(f)}{10}$. Then $\mathbf{Pr}[\mathcal{E}] \geq 1 - \frac{4}{m^2} - \frac{2}{m^{\frac{11}{4}}}$.*

Proof. Let $\mathcal{E}_1$ be the event that the smooth histogram data structure does not fail, $\mathcal{E}_2$ be the event that all instances of CountSketch do not fail, and $\mathcal{E}_3$ be the event that $X \leq \frac{L_2(f)}{10}$ and $\max_{j\in[n]}(Y_j, Z_j) \leq \frac{\alpha L_2(f)}{10}$, so that $\mathcal{E} = \mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3$. It suffices to set the probability of failure $\delta = \frac{1}{m^2}$ in each L_2 estimation algorithm to achieve $\mathbf{Pr}[\mathcal{E}_1] = 1 - \frac{1}{m^2}$. Similarly, by setting the probability of failure $\delta = \frac{1}{m^2}$ in Theorem 2.15, it follows that CountSketch$_a$ succeeds with probability $1 - \frac{1}{m^2}$ and thus $\mathbf{Pr}[\mathcal{E}_2] = 1 - \frac{1}{m^2}$. Finally, note that for window sizes of length W with $W = \mathcal{O}\left(\frac{\log^5 m}{\alpha^2\varepsilon^2}\right)$, we can simply store the entire set of active items. Thus we assume $W = \Omega\left(\frac{\log^5 m}{\alpha^2\varepsilon^2}\right)$ so that $L_2(f) = \Omega\left(\frac{\log^2 m}{\alpha\varepsilon}\right)$. Let $\mathbf{Pr}\left[\widehat{L_2}(f) \leq 2L_2(f)\right] \leq 1/m^2$, then since $X \sim \mathsf{Lap}\left(\frac{1}{40\log m}\widehat{L_2}(f)\right)$ and $Y_k, Z_k \sim \mathsf{Lap}\left(\frac{\alpha}{75\log m}\widehat{L_2}(f)\right)$ for each $k \in [n]$, it follows from Claim 3.4 that $\mathbf{Pr}\left[X \leq \frac{L_2(f)}{10} \wedge \max_{j\in[n]}(Y_j, Z_j) \leq \frac{\alpha L_2(f)}{10}\right] \geq 1 - \left(\frac{2}{m^2} + \frac{2}{m^{\frac{11}{4}}}\right)$. Hence, by a union bound, we have $\mathbf{Pr}[\mathcal{E}] \geq 1 - \frac{4}{m^2} - \frac{2}{m^{\frac{11}{4}}}$. □

Before analyzing the privacy guarantees of Algorithm 5, we must analyze the local sensitivity of its subroutines. We first show a β-smooth upper bound on the local sensitivity of the frequency moment.

Lemma 3.6 (Smooth sensitivity of the frequency moment)**.** *Let $\mathfrak{S}$ be a data stream of length m that induces a frequency vector f and let $\widehat{L_2}(f)$ be the estimate of $L_2(f)$ output by the smooth histogram. Define the function $g(f)$ by*

$$g(f) = \begin{cases} \widehat{L_2}(f), & \text{if } \left(1 - \frac{\varepsilon}{500\log m}\right)L_2(f) \leq \widehat{L_2}(f) \leq \left(1 + \frac{\varepsilon}{500\log m}\right)L_2(f), \\ \left(1 - \frac{\varepsilon}{500\log m}\right)L_2(f), & \text{if } \widehat{L_2}(f) < \left(1 - \frac{\varepsilon}{500\log m}\right)L_2(f), \text{ and} \\ \left(1 + \frac{\varepsilon}{500\log m}\right)L_2(f), & \text{if } \widehat{L_2}(f) > \left(1 + \frac{\varepsilon}{500\log m}\right)L_2(f). \end{cases}$$

Then the function $S(f) = \frac{\varepsilon}{200\log m}\, g(f) + 2$ is a β-smooth upper bound on the local sensitivity of $g(f)$ for $\beta \geq \frac{\varepsilon}{150\log m}$, $\varepsilon > \frac{1000\log m}{\sqrt{W}}$, and sufficiently large W.

Proof. Let $S(f) = \frac{\varepsilon}{200 \log m} g(f) + 2$ and suppose $\|f - f'\|_1 = 1$. Moreover, suppose without loss of generality that $g(f) \geq g(f')$. We first show that Condition 1 of Definition 2.7 holds. The local sensitivity of $g(f)$ is at most

$$|g(f) - g(f')| \leq \left| \left(1 + \frac{\varepsilon}{500 \log m}\right) L_2(f) - \left(1 - \frac{\varepsilon}{500 \log m}\right) L_2(f') \right|.$$

By Lemma 2.13, we have $|L_2(f) - L_2(f')| \leq 2$. Thus for $\varepsilon > \frac{1000 \log m}{\sqrt{W}}$ and sufficiently large W,

$$\begin{aligned}
|g(f) - g(f')| &\leq 2 + \left(\frac{\varepsilon}{500 \log m}\right) L_2(f) + \left(\frac{\varepsilon}{500 \log m}\right) L_2(f') \\
&\leq 2 + \left(\frac{\varepsilon}{500 \log m}\right) (2L_2(f) + 2) \\
&\leq 2 + \left(\frac{\varepsilon}{225 \log m}\right) L_2(f) \\
&\leq 2 + \left(\frac{\varepsilon}{225 \log m}\right) \left(\frac{1}{1 - \varepsilon/(500 \log m)}\right) g(f) \\
&\leq 2 + \frac{\varepsilon}{200 \log m} g(f) = S(f).
\end{aligned}$$

Thus Condition 1 of Definition 2.7 holds. We next show that Condition 2 of Definition 2.7 holds. Furthermore, from the above, we have

$$\begin{aligned}
S(f') &= \frac{\varepsilon}{200 \log m} g(f') + 2 \\
&\leq \left(\frac{\varepsilon}{200 \log m}\right) \left(g(f) + 2 + \left(\frac{\varepsilon}{200 \log m}\right) L_2(f)\right) + 2 \\
&\leq \left(1 + \frac{\varepsilon}{150 \log m}\right) \left(\frac{\varepsilon}{200 \log m} g(f) + 2\right) \\
&\leq e^{\beta} S(f).
\end{aligned}$$

Therefore, both conditions of Definition 2.7 hold and it follows that the function $S(f) = \frac{\varepsilon}{200 \log m} g(f) + 2$ is a β-smooth upper bound on the local sensitivity of $g(f)$ for $\beta \geq \frac{\varepsilon}{150 \log m}$, $\varepsilon > \frac{1000 \log m}{\sqrt{W}}$, and sufficiently large W. □

We next show a β-smooth upper bound on the local sensitivity for each estimated frequency output by Algorithm 5.

Lemma 3.7 (Smooth sensitivity of the estimated frequency)**.** *Let $\mathfrak{S}$ be a data stream of length m that induces a frequency vector f and let $\widehat{f}_k$ be the estimate of the frequency of a coordinate $k \in [n]$ output by the smooth histogram. Define the function $h(f)$ by*

$$h(f) = \begin{cases} \widehat{f}_k, & \text{if } f_k - \frac{\alpha^3 \varepsilon}{1000 \log m} L_2(f) \leq \widehat{f}_k \leq f_k + \frac{\alpha^3 \varepsilon}{1000 \log m} L_2(f), \\ f_k - \frac{\alpha^3 \varepsilon}{1000 \log m} L_2(f), & \text{if } \widehat{f}_k < f_k - \frac{\alpha^3 \varepsilon}{1000 \log m} L_2(f), \text{ and} \\ f_k + \frac{\alpha^3 \varepsilon}{1000 \log m} L_2(f), & \text{if } \widehat{f}_k > f_k + \frac{\alpha^3 \varepsilon}{1000 \log m} L_2(f). \end{cases}$$

Then the function $S(f) = \frac{\alpha^3 \varepsilon}{200 \log m} h(f) + 2$ is a β-smooth upper bound on the local sensitivity of $h(f)$ for $\beta \geq \frac{\alpha^3 \varepsilon}{150 \log m}$, $\varepsilon > \frac{1000 \log m}{\sqrt{W} \alpha^3}$, and sufficiently large W.

Proof. The proof is almost identical to that of Lemma 3.6; we include it for completeness. Let $S(f) = \frac{\alpha^3\varepsilon}{200\log m}\, h(f) + 2$ and suppose $\|f - f'\|_1 = 1$. Moreover, suppose without loss of generality that $h(f) \geq h(f')$. We first show that Condition 1 of Definition 2.7 holds. The local sensitivity of $h(f)$ is at most

$$|h(f) - h(f')| \leq |f_k - f'_k| + \frac{\alpha^3\varepsilon}{1000\log m}\,(L_2(f) + L_2(f')).$$

By Lemma 2.13, we have $|L_2(f) - L_2(f')| \leq 2$. Since $|f_k - f'_k| \leq 2$, we further have for $\varepsilon > \frac{1000\log m}{\sqrt{W}\alpha^3}$ and sufficiently large W,

$$\begin{aligned}
|h(f) - h(f')| &\leq 2 + \frac{\alpha^3\varepsilon}{1000\log m}\,(L_2(f) + L_2(f) + 2)\\
&\leq 2 + \left(\frac{\alpha^3\varepsilon}{1000\log m}\right)(2L_2(f) + 2)\\
&\leq 2 + \left(\frac{\alpha^3\varepsilon}{400\log m}\right) L_2(f)\\
&\leq 2 + \left(\frac{\alpha^3\varepsilon}{400\log m}\right)\left(\frac{1}{1 - (\alpha^3\varepsilon)/(1000\log m)}\right) h(f)\\
&\leq 2 + \frac{\alpha^3\varepsilon}{200\log m}\, h(f) = S(f).
\end{aligned}$$

Thus Condition 1 of Definition 2.7 holds. We next show that Condition 2 of Definition 2.7 holds. Furthermore, from the above, we have

$$\begin{aligned}
S(f') &= \frac{\alpha^3\varepsilon}{200\log m}\, h(f') + 2\\
&\leq \left(\frac{\alpha^3\varepsilon}{200\log m}\right)\left(h(f) + 2 + \left(\frac{\alpha^3\varepsilon}{400\log m}\right) L_2(f)\right) + 2\\
&\leq \left(1 + \frac{\alpha^3\varepsilon}{150\log m}\right)\left(\frac{\alpha^3\varepsilon}{200\log m}\, h(f) + 2\right)\\
&\leq e^{\beta}\, S(f).
\end{aligned}$$

Therefore, both conditions of Definition 2.7 hold and it follows that the function $S(f) = \frac{\alpha^3\varepsilon}{200\log m}\, h(f) + 2$ is a β-smooth upper bound on the local sensitivity of $h(f)$ for $\beta \geq \frac{\alpha^3\varepsilon}{150\log m}$, $\varepsilon > \frac{1000\log m}{\sqrt{W}\alpha^3}$, and sufficiently large W. □

With the structural results on smooth sensitivity in place, we now show that Algorithm 5 is (ε, δ)-differentially private.

Lemma 3.8. *There exists an algorithm (see Algorithm 5) that is (ε, δ)-differentially private for $\alpha \in (0, 1)$, $\varepsilon > \frac{1000\log m}{\sqrt{W}\alpha^3}$, and $\delta > \frac{6}{m^2}$.*

Proof. Let $\mathcal{E}$ be the event that (1) the smooth histogram data structure does not fail on either stream, (2) all instances of CountSketch do not fail, and (3)$X \leq \frac{L_2(f)}{10}$ and $\max_{j\in[n]}(Y_j, Z_j) \leq \frac{\alpha L_2(f)}{10}$. By Lemma 3.5, we have $\mathbf{Pr}\left[\mathcal{E}\right] \geq 1 - \frac{4}{m^2} - \frac{2}{m^{11/4}}$. Conditioned on $\mathcal{E}$, then by Lemma 3.3, the algorithm releases at most $\frac{10}{\alpha^2}$ frequencies.

Let f be the frequency vector defined by the active elements of the sliding window. Let $g(f)$ be the function defined in Lemma 3.6. Then by Lemma 3.6, we have that $\frac{\varepsilon}{125\log m}\, g(f)$ is a β-smooth upper bound on the local sensitivity of $g(f)$, for $\beta = \frac{\varepsilon}{2\ln(2m^3)}$ and sufficiently large m. Thus by Theorem 2.8 it suffices to add Laplacian noise $X \sim \mathsf{Lap}\left(\frac{1}{40}\, g(f)\right)$ to the estimated norm $\widehat{L_2}$ output by the smooth histogram to achieve $\left(\frac{\varepsilon}{3}, \frac{\delta_1}{3}\right)$-differential privacy where $\frac{\delta_1}{3} = \frac{1}{m^3}$.

Similarly by Lemma 3.7, we have that $\frac{\alpha^3\varepsilon}{200\log m}\, h(f)$ is a β-smooth upper bound on the local sensitivity of each estimated frequency for $\beta = \frac{\varepsilon}{2\ln(2m^3)}$ and sufficiently large m. Thus by Theorem 2.8 it suffices to add Laplacian noise $Y_k \sim \mathsf{Lap}\left(\frac{\alpha}{75\log m}\, h(f)\right)$ to each estimated frequency $\widehat{f_k}$ output by the smooth histogram to achieve $\left(\frac{\varepsilon\alpha^2}{30}, \frac{\delta_2}{3}\right)$-differential privacy where $\frac{\delta_2}{3} = \frac{1}{m^3}$. Since we only release at most $\frac{10}{\alpha^2}$ estimated frequencies, then by Theorem 2.5 and Theorem 2.9, the estimated frequencies for each heavy-hitter are $(\frac{\varepsilon}{3}, \frac{\delta_2}{3})$-differentially private. Finally, we only release the estimated frequencies that are above a threshold $\frac{3\alpha}{4}\left(\widehat{L_2} + X\right) + Y_k$, which corresponds to a limited histogram query. Hence by Theorem 2.9, the algorithm would be $(\varepsilon, \frac{2\delta}{3})$-differentially private when using the functions $g(f)$, to which we do not have access. Instead, we note that conditioned on $\mathcal{E}$, we have that $g(f) = \widehat{L_2}(f)$ with probability $1 - \frac{4}{m^2} - \frac{2}{m^{\frac{11}{4}}}$. Setting $\frac{\delta_3}{3} = \frac{5}{m^2} > \frac{4}{m^2} - \frac{2}{m^{\frac{11}{4}}}$, and consequently, setting $\delta = \frac{6}{m^2} > \frac{\delta_1}{3} + \frac{\delta_2}{3} + \frac{\delta_3}{3} = \frac{2}{m^3} + \frac{5}{m^2}$, this is absorbed into the failure probability. Thus the entire algorithm is (ε, δ)-differentially private. □

Since Algorithm 5 further adds Laplacian noise $Z \sim \mathsf{Lap}\left(\frac{\alpha}{75\log m}\, \widehat{L_2}(f)\right)$ to each $\widehat{f_k}$ with $k \in \mathcal{L}$, then Lemma 3.3 implies that the additive error to f_k is $\frac{\alpha}{50\log m}\, L_2(f) + \mathsf{Lap}\left(\frac{\alpha}{75\log m}\, \widehat{L_2}(f)\right)$ for each reported coordinate $k \in [n]$.

Thus we now have our main result for differentially private L_2-heavy hitters in the sliding window model.

Theorem 1.2. *For any $\alpha \in (0,1), c > 0$, window parameter W on a stream of length m that induces a frequency vector $f \in \mathbb{R}^n$ in the sliding window model, and privacy parameter $\varepsilon > \frac{1000\log m}{\alpha^3\sqrt{W}}$, there exists an algorithm such that:*

(1) (Privacy) The algorithm is (ε, δ)-differentially private for $\delta = \frac{1}{m^c}$.

(2) (Heavy-hitters) With probability at least $1 - \frac{1}{m^c}$, the algorithm outputs a list $\mathcal{L}$ such that $k \in \mathcal{L}$ for each $k \in [n]$ with $f_k \geq \alpha\, L_2(f)$ and $j \notin \mathcal{L}$ for each $j \in [n]$ with $f_j \leq \frac{\alpha}{2}\, L_2(f)$.

(3) (Accuracy) With probability at least $1 - \frac{1}{m^c}$, we simultaneously have $|f_k - \widetilde{f_k}| \leq \frac{\alpha}{4}\, L_2(f)$ for all $k \in \mathcal{L}$, where $\widetilde{f_k}$ denotes the noisy approximation of f_k output by the algorithm.

(4) (Complexity) The algorithm uses $\mathcal{O}\left(\frac{\log^7 m}{\alpha^6\eta^4}\right)$ bits of space and $\mathcal{O}\left(\frac{\log^4 m}{\alpha^3\eta^4}\right)$ operations per update where $\eta = \max\{1, \varepsilon\}$.

Proof. Let $\mathcal{E}$ be the event that (1) the smooth histogram data structure does not fail on either stream, (2) all instances of CountSketch do not fail, and (3) $\max_{j\in[n]}(X, Y_j, Z_j) \leq \frac{\alpha L_2(f)}{10}$. By Lemma 3.5, we have $\mathbf{Pr}\left[\mathcal{E}\right] \geq 1 - \frac{1}{m^c}$. Conditioned on $\mathcal{E}$, the first and second claims follow immediately from Lemma 3.2 and Lemma 3.3. The third claim follows from Lemma 3.8.

To analyze the space complexity, note that by Theorem 2.12, each $\left(1+\frac{\varepsilon}{500\log m}\right)$-approximation algorithm for L_2 with failure probability $\delta = \frac{1}{m^c}$ uses $\mathcal{O}\left(\frac{1}{\varepsilon^2}\log^2 m\right)$ bits of space. Similarly by Theorem 2.15, each instance of CountSketch with threshold $\frac{\alpha^3\varepsilon}{500\log m}$ uses $\mathcal{O}\left(\frac{1}{\alpha^6\varepsilon^2}\log^4 m\right)$ bits of space. By the smooth histogram data structure, the timestamps $t_1,\ldots,t_s$ satisfy $L_2(t_i,m) \geq \left(1+\frac{\varepsilon}{500\log m}\right) L_2(t_{i+2},m)$. By the smoothness of L_2 in Lemma 2.19, this requires the invariant that $L_2(t_i,t) \leq \left(1+\left(\frac{\varepsilon}{1000\log m}\right)^2\right) L_2(t_{i+1},t)$ for all $t\in[m]$. Since the L_2 of the entire stream is at most m, $s = \mathcal{O}\left(\frac{1}{\varepsilon^2}\log^3 m\right)$. Thus there are at most $\mathcal{O}\left(\frac{1}{\varepsilon^2}\log^3 m\right)$ instances of CountSketch and AMS. Hence the total space used by CountSketch and AMS is $\mathcal{O}\left(\frac{1}{\alpha^6\varepsilon^4}\log^7 m\right)$. By Lemma 3.1, the total space used by the frequency estimation algorithms Counter across all $\mathcal{O}\left(\frac{1}{\varepsilon}\log m\right)$ instances is also at most $\mathcal{O}\left(\frac{1}{\alpha^6\varepsilon^4}\log^7 m\right)$. Therefore, the total space used by the algorithm is $\mathcal{O}\left(\frac{1}{\alpha^6\varepsilon^4}\log^7 m\right)$ bits of space.

To analyze the time complexity, note that by Theorem 2.15 and Lemma 3.1, $\mathcal{O}\left(\frac{\log n}{\alpha^2\varepsilon^2}+\log m\right)$ operations are required per update, per instance of CountSketch and Counter. The update time then follows from the fact that there are $\mathcal{O}\left(\frac{1}{\varepsilon^2}\log^3 m\right)$ simultaneous instances. □

Finally, observe that in the proof of Lemma 3.8, each instance of the frequency estimation algorithm is required to be $\left(\frac{\varepsilon}{30\alpha^2},\frac{\delta}{3}\right)$-differentially private, since we release $\mathcal{O}\left(\frac{1}{\alpha^2}\right)$ frequencies, by the standard differential privacy composition theorem, i.e., Theorem 2.9. We can instead use advanced composition, i.e., Theorem 2.10, to require each instance of the frequency estimation algorithm to be $\left(\mathcal{O}\left(\frac{\varepsilon}{\alpha\sqrt{\log m}}\right),\frac{\delta}{3}\right)$-differentially private, for $\varepsilon = \mathcal{O}\left(\alpha^2\right)$. In this case, we can require each frequency estimation algorithm to have a cruder approximation guarantee, e.g., using CountSketch with threshold $\mathcal{O}\left(\frac{\alpha^2\varepsilon}{\log^{3/2} m}\right)$ rather than threshold $\frac{\alpha^3\varepsilon}{500\log m}$.

Theorem 3.9. *For any $\alpha, c > 0$, window parameter W on a stream of length m that induces a frequency vector $f\in\mathbb{R}^n$ in the sliding window model and privacy parameter ε, there exists an algorithm (see Algorithm 5) such that:*

(1) With probability at least $1-\frac{1}{m^c}$, the algorithm outputs a list $\mathcal{L}$ such that $k\in\mathcal{L}$ for each $k\in[n]$ with $f_k \geq \alpha\, L_2(f)$ and $j\notin\mathcal{L}$ for each $j\in[n]$ with $f_j \leq \frac{\alpha}{2}\, L_2(f)$.

(2) With probability at least $1-\frac{1}{\mathrm{poly}(m)}$, we simultaneously have $|f_k-\widetilde{f}_k| \leq \frac{\alpha}{4}\, L_2(f)$ for all $k\in\mathcal{L}$, where $\widetilde{f}_k$ denotes the noisy approximation of f_k output by Algorithm 5.

(3) The algorithm is (ε,δ)-differentially private for $\delta = \frac{1}{m^c}$.

(4) The algorithm uses $\mathcal{O}\left(\frac{\log^8 m}{\alpha^4\eta^3}\right)$ bits of space and $\mathcal{O}\left(\frac{\log^{9/2} m}{\alpha^2\eta^3}\right)$ operations per update where $\eta = \max\{1,\varepsilon\}$.

4 Better L_1 Heavy-Hitter Algorithm

In this section, we show how our main technical ideas can be simplified and applied to give a better L_1-heavy hitter algorithm than the one-shot sliding window algorithm given in [Upa19]. Due to the linearity of L_1, our algorithm for differentially private L_1-heavy hitters in the sliding

window model is significantly simpler than the L_2-heavy hitters algorithm. For starters, each set of $c > 0$ updates must contribute exactly c to the L_1 norm, whereas their contribution to the L_2 norm depends on the particular coordinates they update. Therefore, not only do we not require an algorithm to approximate the L_1 norm of the active elements of the sliding window, but also we can fix a set of static timestamps in the smooth histogram, so we do not need to perform the same analysis to circumvent the sensitivity of the timestamps. Instead, it suffices to initialize a *deterministic* L_1-heavy hitter algorithm at each timestamp and maintain deterministic counters for each reported heavy-hitter. Pure differential privacy then follows from the lack of failure conditions in the subroutines, which was not possible for L_2-heavy hitters.

We first require the following *deterministic* algorithm for L_1-heavy hitters.

Theorem 4.1 (Heavy-hitter algorithm MisraGries)**.** *[MG82] Given an accuracy parameter $\alpha > 0$, there exists a one-pass algorithm* MisraGries *for the L_1-heavy hitter problem, using $\mathcal{O}\left(\frac{1}{\alpha}\log m\right)$ bits of space and update time on a stream of length m and a universe of size n.*

The global L_1-sensitivity of the MisraGries algorithm is upper bounded as follows:

Fact 4.2. MisraGries *on a stream of length m and threshold α has L_1 sensitivity αm.*

The algorithm for differentially private L_1-heavy hitters in the sliding window model is much simpler than the L_2-heavy hitters algorithm due to the linearity of L_1. For instance, because we know that each set of c updates contributes exactly c to the L_1 norm, then it suffices to maintain a static set of timestamps in the smooth histogram. Therefore, the timestamps do not change across neighboring datasets, so we do not need to analyze the sensitivity of the timestamps. Similarly, we do not need an algorithm to approximate the L_1 norm of the substream starting at each timestamp, since it is also fixed. Hence, we do not need analysis for the smooth sensitivity of an L_1 norm approximation algorithm and the subsequent statistical distance analysis that we needed for AMS for L_2 norm approximation. Instead, it suffices to maintain a counter for each reported heavy-hitter by some instance of MisraGries starting at each of the timestamps. We then output the coordinates whose estimated frequencies by the counters surpass a fixed threshold in terms of W, which again is deterministic due to the linearity of L_1. We give the algorithm in full in Algorithm 6.

We first analyze the accuracy of the estimated frequency for each item output by Algorithm 6.

Lemma 4.3 (Accuracy of frequency estimation)**.** *Let $\mathcal{E}$ be the event that $\max_{j\in[n]}(Z_j) \le \frac{\alpha}{16}W$. Then for each $k \in [n]$ with $f_k \ge \alpha\, L_1(f)$, we have $k \in \mathcal{L}$. Moreover, for each item $k \in \mathcal{L}$,*

$$|f_k - \widetilde{f}_k| \le \frac{\alpha}{8}W.$$

Proof. Consider a time $t \in [m]$ that induces a frequency vector f. We have $L_1(f) \le L_1(t_a : t) \le \left(1 + \frac{1}{100}\right) L_1(f)$. Hence for each $k \in [n]$ with $f_k \ge \alpha\, L_1(f)$, we have $f_k \ge \frac{\alpha}{16}\, L_1(t_a : t)$ and thus k will be reported by the instance of MisraGries starting at time t_a. For each item k reported by MisraGries starting at time t_a, Counter reports an estimated frequency $\widehat{f}_k$ such that

$$|f_k - \widehat{f}_k| \le \frac{\alpha}{32}\, L_1(t_a : t) \le \frac{\alpha}{16}\, W.$$

Conditioned on the event that $\max_{j\in[n]}(Z_j) \le \frac{\alpha}{16}W$, then we have $|\widetilde{f}_k - \widehat{f}_k| \le \frac{\alpha}{16}W$. Hence by triangle inequality,

$$|f_k - \widetilde{f}_k| \le \frac{\alpha}{16}\, L_1(t_a : t) \le \frac{\alpha}{8}\, W$$

for each item $k \in \mathcal{L}$. □

Algorithm 6 Differentially private sliding window algorithm for L_1-heavy hitters

Input: Stream $\mathfrak{S}$, threshold/accuracy parameter $\alpha \in (0,1)$, differential privacy parameters $\varepsilon, \delta > 0$, window parameter $W > 0$, size n of the underlying universe, upper bound m on stream length

Output: A list $\mathcal{L}$ of L_1 heavy-hitters with approximate frequencies

Process the stream $\mathfrak{S}$, maintaining timestamps $t_1, \ldots, t_s$ at each time $t \in [m]$ so that for each $i \in [s]$, either $i = 1$, $i = s$, or $t - t_{i-1} + 1 > \left(1 + \frac{1}{100}\right)(t - t_{i+1} + 1)$ through a smooth histogram

Implement heavy-hitter algorithm MisraGries on the substream starting at t_i for each $i \in [s]$ with threshold $\frac{\alpha}{16}$

Set $a = \max\{i \in [s] : t_i \leq m - W + 1\}$ on window query $W > 0$

for each heavy-hitter $k \in [n]$ reported by MisraGries starting at t_a **do**

    Run Counter with additive error $\frac{\alpha}{32}(t - t_a + 1)$

    Let $\widehat{f}_k$ be the approximate frequency reported by Counter

    $Z_k \leftarrow \mathsf{Lap}\left(\frac{1}{\varepsilon\alpha \log m}\right)$, $\widetilde{f}_k = \widehat{f}_k + Z_k$

    if $\widetilde{f}_k \geq \frac{3\alpha}{4} W$ **then**

        $\mathcal{L} \leftarrow \mathcal{L} \cup \{(k, \widetilde{f}_k)\}$

return $\mathcal{L}$

We now show that Algorithm 6 guarantees pure differential privacy. By comparison, the algorithm of [Upa19] can only guarantee (ε, δ)-differential privacy due to the failure probability of their randomized subroutines.

Lemma 4.4. *Algorithm 6 is $(\varepsilon, 0)$-differentially private for any constant $\varepsilon \in (0, 1]$.*

Proof. Note that unlike for the L_2-heavy hitter algorithm where the times $t_1, \ldots, t_s$ are determined by the output of the randomized algorithms for L_2-norm estimation, the times $t_1, \ldots, t_s$ are deterministic since the L_1 norm of the frequency vector defined from a time t to a time t' is simply $t' - t + 1$. Thus the sensitivity of the estimated frequency of each element is the sensitivity of the instance of MisraGries that starts at time t_a. Consequently by Fact 4.2, the sensitivity of each estimated frequency is at most 2. Hence by Theorem 2.5, it suffices to add Laplacian noise that scales with $\frac{2}{\varepsilon}$ to each estimated frequency to preserve $(\varepsilon, 0)$-differential privacy. □

We now give the full guarantees of Algorithm 6.

Theorem 4.5. *For any $\alpha, c > 0$, window parameter W on a stream of length m that induces a frequency vector $f \in \mathbb{R}^n$ in the sliding window model and constant privacy parameter ε, there exists an algorithm (see Algorithm 6) such that:*

(1) With probability at least $1 - \frac{1}{m^c}$, the algorithm outputs a list $\mathcal{L}$ such that $k \in \mathcal{L}$ for each $k \in [n]$ with $f_k \geq \alpha W$ and $j \notin \mathcal{L}$ for each $j \in [n]$ with $f_j \leq \frac{\alpha}{2} W$.

(2) With probability at least $1 - \frac{1}{m^c}$, we simultaneously have $|f_k - \widetilde{f}_k| \leq \frac{\alpha}{4} W$ for all $k \in \mathcal{L}$ where $\widetilde{f}_k$ denotes the noisy approximation of f_k output by Algorithm 6.

(3) The algorithm is $(\varepsilon, 0)$-differentially private.

(4) The algorithm uses $\mathcal{O}\left(\frac{\log^2 m}{\alpha}\right)$ bits of space and $\mathcal{O}\left(\frac{\log^2 m}{\alpha}\right)$ update time per arriving symbol.

Proof. The first three statements follow from Lemma 4.3 and Lemma 4.4. Thus it remains to analyze the space complexity. Note that for each time t_i with $i \in [s]$, we have either $i = 1$, $i = s$, or $t - t_{i-1} + 1 > \left(1 + \frac{1}{100}\right)(t - t_{i+1} + 1)$, so that $s = \mathcal{O}(\log m) = \mathcal{O}(\log n)$. Each time t_i corresponds to a separate instance of MisraGries that uses $\mathcal{O}\left(\frac{1}{\alpha}\log n\right)$ bits of space and update time per arriving symbol. The instances of Counter associated with each time t_i can also maintain $\mathcal{O}\left(\frac{1}{\alpha}\right)$ times, which use $\mathcal{O}\left(\frac{1}{\alpha}\log m\right)$ space and update time for each time t_i. Hence, the total space and update time is $\mathcal{O}\left(\frac{\log^2 m}{\alpha}\right)$. □

References

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999. 9

[AS19] Jayadev Acharya and Ziteng Sun. Communication complexity in locally private distribution estimation and heavy hitters. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, pages 51–60, 2019. 5

[BBDS12] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The johnson-lindenstrauss transform itself preserves differential privacy. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 410–419, 2012. 2, 4

[BCIW16] Vladimir Braverman, Stephen R. Chestnut, Nikita Ivkin, and David P. Woodruff. Beating countsketch for heavy hitters in insertion streams. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 740–753, 2016. 9

[BDB16] Jeremiah Blocki, Anupam Datta, and Joseph Bonneau. Differentially private password frequency lists. In *23rd Annual Network and Distributed System Security Symposium, NDSS*. The Internet Society, 2016. 2

[BDM+20] Vladimir Braverman, Petros Drineas, Cameron Musco, Christopher Musco, Jalaj Upadhyay, David P. Woodruff, and Samson Zhou. Near optimal linear algebra in the online and sliding window models. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 517–528, 2020. 3, 4

[BDN17] Jaroslaw Blasiok, Jian Ding, and Jelani Nelson. Continuous monitoring of ℓ_p norms in data streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 32:1–32:13, 2017. 9

[BEL+20] Michele Borassi, Alessandro Epasto, Silvio Lattanzi, Sergei Vassilvitskii, and Morteza Zadimoghaddam. Sliding window algorithms for k-clustering problems. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020. 3, 4

[BGL+18] Vladimir Braverman, Elena Grigorescu, Harry Lang, David P. Woodruff, and Samson Zhou. Nearly optimal distinct elements and heavy hitters on sliding windows. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 7:1–7:22, 2018. 3, 4, 5, 6

[BGO14] Vladimir Braverman, Ran Gelles, and Rafail Ostrovsky. How to catch L_2-heavy-hitters on sliding windows. *Theor. Comput. Sci.*, 554:82–94, 2014. 3, 4, 5, 6

[BJKS04] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004. 3

[BLLM16] Vladimir Braverman, Harry Lang, Keith Levin, and Morteza Monemizadeh. Clustering problems on sliding windows. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1374–1390, 2016. 3, 4

[BLUZ19] Vladimir Braverman, Harry Lang, Enayat Ullah, and Samson Zhou. Improved algorithms for time decay streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 27:1–27:17, 2019. 2

[BNS19] Mark Bun, Jelani Nelson, and Uri Stemmer. Heavy hitters and the structure of local privacy. *ACM Trans. Algorithms*, 15(4):51:1–51:40, 2019. 4, 5

[BNST20] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Thakurta. Practical locally private heavy hitters. *J. Mach. Learn. Res.*, 21:16:1–16:42, 2020. 4, 5

[BO07] Vladimir Braverman and Rafail Ostrovsky. Smooth histograms for sliding windows. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS), Proceedings*, pages 283–293, 2007. 3, 4, 6

[BO10] Vladimir Braverman and Rafail Ostrovsky. Effective computations on sliding windows. *SIAM J. Comput.*, 39(6):2113–2131, 2010. 11, 12

[BS80] Jon Louis Bentley and James B. Saxe. Decomposable searching problems I: static-to-dynamic transformation. *J. Algorithms*, 1(4):301–358, 1980. 3

[BS15] Raef Bassily and Adam D. Smith. Local, private, efficient protocols for succinct histograms. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC*, pages 127–135, 2015. 4, 5

[BWZ21] Vladimir Braverman, Viska Wei, and Samson Zhou. Symmetric norm estimation and regression on sliding windows. In *Computing and Combinatorics - 27th International Conference, COCOON, Proceedings*, pages 528–539, 2021. 3

[CCF04] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004. 10

[CKS03] Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *18th Annual IEEE Conference on Computational Complexity*, pages 107–117, 2003. 3

[CLSX12] T.-H. Hubert Chan, Mingfei Li, Elaine Shi, and Wenchang Xu. Differentially private continual monitoring of heavy hitters from distributed streams. In *Privacy Enhancing Technologies - 12th International Symposium, PETS. Proceedings*, pages 140–159, 2012. 3, 5

[CNZ16] Jiecao Chen, Huy L. Nguyen, and Qin Zhang. Submodular maximization over sliding windows. *CoRR*, abs/1611.00129, 2016. 4

[CS06] Edith Cohen and Martin J. Strauss. Maintaining time-decaying stream aggregates. *J. Algorithms*, 59(1):19–36, 2006. 2

[CSS11] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):26:1–26:24, 2011. 3

[DGIM02] Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM J. Comput.*, 31(6):1794–1813, 2002. 3, 4

[DKY17] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 3571–3580, 2017. 5

[DMNS16] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. *J. Priv. Confidentiality*, 7(3):17–51, 2016. 1, 2

[DNP+10] Cynthia Dwork, Moni Naor, Toniann Pitassi, Guy N. Rothblum, and Sergey Yekhanin. Pan-private streaming algorithms. In *Innovations in Computer Science - ICS. Proceedings*, pages 66–80, 2010. 3, 4

[DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014. 8, 9

[Dwo06] Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP, Proceedings, Part II*, pages 1–12, 2006. 1

[ELVZ17] Alessandro Epasto, Silvio Lattanzi, Sergei Vassilvitskii, and Morteza Zadimoghaddam. Submodular optimization over sliding windows. In *Proceedings of the 26th International Conference on World Wide Web, WWW*, pages 421–430, 2017. 4

[EMMZ21] Alessandro Epasto, Mohammad Mahdian, Vahab S. Mirrokni, and Peilin Zhong. Improved sliding window algorithms for clustering and coverage via bucketing-based sketches. *CoRR*, abs/2110.15533, 2021. 4

[EPK14] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 1054–1067, 2014. 1

[Fac] Facebook. https://www.facebook.com/policy.php. 1

[Goo] Google. https://policies.google.com/technologies/retention. 1

[Gre16] Andy Greenberg. Apple's 'differential privacy'is about collecting your data—but not your data. *Wired, June*, 13, 2016. 1

[HKM+20] Avinatan Hassidim, Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. Adversarially robust streaming algorithms via differential privacy. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020. 2

[HQYC21] Ziyue Huang, Yuan Qiu, Ke Yi, and Graham Cormode. Frequency estimation under multiparty differential privacy: One-shot and streaming. *CoRR*, abs/2104.01808, 2021. 2, 3, 4

[JRUW20] Matthew Joseph, Aaron Roth, Jonathan R. Ullman, and Bo Waggoner. Local differential privacy for evolving data. *J. Priv. Confidentiality*, 10(1), 2020. 2

[JWZ21] Rajesh Jayaram, David P. Woodruff, and Samson Zhou. Truly perfect samplers for data streams and sliding windows. *CoRR*, abs/2108.12017, 2021. 3, 4

[KP08] Tsvi Kopelowitz and Ely Porat. Improved algorithms for polynomial-time decay and time-decay with additive error. *Theory Comput. Syst.*, 42(3):349–365, 2008. 2

[LMWY20] Kasper Green Larsen, Tal Malkin, Omri Weinstein, and Kevin Yeo. Lower bounds for oblivious near-neighbor search. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1116–1134, 2020. 3

[MG82] Jayadev Misra and David Gries. Finding repeated elements. *Sci. Comput. Program.*, 2(2):143–152, 1982. 22

[MMNW11] Darakhshan J. Mir, S. Muthukrishnan, Aleksandar Nikolov, and Rebecca N. Wright. Pan-private algorithms via statistics on sketches. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*, pages 37–48, 2011. 2

[NRS07] Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 75–84, 2007. 6, 8

[NXY+16] Thông T. Nguyên, Xiaokui Xiao, Yin Yang, Siu Cheung Hui, Hyejin Shin, and Junbum Shin. Collecting and analyzing data from smart device users with local differential privacy. *CoRR*, abs/1606.05053, 2016. 1

[SYC18] Shang-Yu Su, Pei-Chieh Yuan, and Yun-Nung Chen. How time matters: Learning time-decay attention for contextual spoken language understanding in dialogues. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 2133–2142, 2018. 2

[Upa19] Jalaj Upadhyay. Sublinear space private algorithms under the sliding window model. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, pages 6363–6372, 2019. 1, 3, 4, 5, 6, 21, 23, 30

[UU21] Jalaj Upadhyay and Sarvagya Upadhyay. A framework for private matrix analysis in sliding window model. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, pages 10465–10475, 2021. 3, 4

[WZ21] David P. Woodruff and Samson Zhou. Tight bounds for adversarially robust streams and sliding windows via difference estimators. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 1183–1196. IEEE, 2021. 3, 4

A Continual Release of Heavy-Hitters

Our algorithm consists of $L := \mathcal{O}(\log W) = \mathcal{O}(\log n)$ levels of subroutines. In each level $\ell \in [L]$, we split the stream into continuous blocks of length $S_\ell := 2^{\ell-2} \cdot \frac{\alpha\sqrt{W}}{100 \log W}$. Given a threshold parameter $\alpha > 0$, for each block in level ℓ, we run an instance of MisraGries with threshold $\frac{1}{2^{\ell+1}L}$. At the end of the stream, we stitch together a sketch of the underlying dataset represented by the sliding window through a binary tree mechanism.

Algorithm 7 Differentially private sliding window algorithm for heavy-hitters

Input: Stream $\mathfrak{S}$ of length m, accuracy parameter $\alpha > 0$, differential privacy parameter $\varepsilon > 0$, window parameter $W > 0$

Output: Continuous release of list $\mathcal{L}$ of L_1 heavy-hitters with approximate frequencies

$L \leftarrow \left\lceil \log \frac{100\varepsilon}{\alpha} \sqrt{W} + \log\log W \right\rceil + 2$, $\varphi \leftarrow \min(\varepsilon, 1)$
for $\ell \in [L]$ **do**
  Partition the stream into blocks of length $S_\ell := 2^{\ell-2} \cdot \frac{\alpha\sqrt{W}}{100 \log W}$
  Run MisraGries on each block of length S_ℓ with threshold $\frac{\varphi\alpha\sqrt{W}}{16 S_\ell \cdot \log^3 W}$
for each time $i \in [m]$ **do**
  Update each MisraGries algorithm
  Build a binary tree mechanism and output the reported heavy-hitters
  for each active block $B_1, \ldots, B_L$ in the binary tree mechanism **do**
    Set $\widehat{g_i^{(r)}}$ be the estimate of each $i \in [n]$ in B_r
    $\widehat{f_i^{(r)}} \leftarrow \widehat{g_i^{(r)}} + \mathsf{Lap}\left(\frac{\alpha\sqrt{W}}{16 \log^2 W}\right)$
  return $\widehat{f}_i = \sum_{r=1}^{L} \widehat{f_i^{(r)}}$

From Theorem 4.1, we have the following accuracy guarantees:

Corollary A.1. *Suppose* MisraGries *at level $\ell \in [L]$ on a substream induced by the updates between $\left[t+1, t+2^{\ell-2} \cdot \frac{\sqrt{W}}{100 \log W}\right]$ outputs an estimate $\widehat{g}_i$ for the frequency g_i of item $i \in [n]$ between $\left[t+1, t+2^{\ell-2} \cdot \frac{\sqrt{W}}{100 \log W}\right]$. Then $|g_i - \widehat{g}_i| \le \frac{\alpha\sqrt{W}}{16 \log^3 W}$.*

Lemma A.2 (Accuracy of frequencies)**.** *For all $i \in [n]$, we have that $|f_i - \widehat{f}_i| \le \frac{\alpha\sqrt{W}}{8}$.*

Proof. Let t be the smallest multiple of $\frac{\alpha\sqrt{W}}{200 \log W}$ with $t \ge m - W + 1$. In other words, t is the first starting location of a block at level 1 among the active items. For each $i \in [n]$, let $\tilde{f}_i$ be the frequency of i from time t to m. Since $(m - W + 1) - t < \frac{\alpha\sqrt{W}}{200 \log W}$, then we have

$$f_i - \tilde{f}_i < \frac{\alpha\sqrt{W}}{200 \log W}$$

for all $i \in [n]$.

Our binary tree mechanism partitions the stream from time t to m into at most two disjoint blocks in each level. Suppose these blocks are $B_1, \ldots, B_L$ across the L levels. For each $i \in [n]$ and $r \in [L]$, let $g_i^{(r)}$ be the frequency of i within the substream allocated to B_r. Similarly, let $\widehat{g_i^{(r)}}$ be the estimated frequency of i within the substream by the corresponding instance of MisraGries. By Corollary A.1, we have that $|g_i^{(r)} - \widehat{g_i^{(r)}}| < \frac{\alpha\sqrt{W}}{16\log^3 W}$. Thus we have by triangle inequality,

$$\begin{aligned}\left|\tilde{f}_i - \sum_{r=1}^{L} \widehat{g_i^{(r)}}\right| &= \left|\sum_{r=1}^{L} g_i^{(r)} - \sum_{r=1}^{L} \widehat{g_i^{(r)}}\right| \\ &\le \sum_{r=1}^{L} \left|g_i^{(r)} - \widehat{g_i^{(r)}}\right| \le \sum_{r=1}^{L} \frac{\alpha\sqrt{W}}{16\log W} \\ &\le \frac{L\alpha\sqrt{W}}{16\log W} \le \frac{\alpha\sqrt{W}}{16}\end{aligned}$$

Hence by another triangle inequality,

$$\left|f_i - \sum_{r=1}^{L} \widehat{g_i^{(r)}}\right| \le \left|\tilde{f}_i - \sum_{r=1}^{L} \widehat{g_i^{(r)}}\right| + |f_i - \tilde{f}_i| \le \frac{\alpha\sqrt{W}}{8}.$$

The claim then follows from setting $\widehat{f_i} = \sum_{r=1}^{L} \widehat{g_i^{(r)}}$. □

Theorem A.3. *Given threshold/accuracy parameter $\alpha > 0$, privacy parameter $\varepsilon = \mathcal{O}(1)$, and window parameter W on a stream of length m that induces a frequency vector $f \in \mathbb{R}^n$ in the sliding window model, there exists an algorithm such that:*

(1) The algorithm continually outputs a list $\mathcal{L}$ such that $k \in \mathcal{L}$ for each $k \in [n]$ with $f_k \ge \alpha\sqrt{W}$ and with high probability, we have $|f_k - \widehat{f_k}| \le \frac{\alpha\sqrt{W}}{2}$ for each $k \in \mathcal{L}$.

(2) The algorithm is $(\varepsilon, 0)$-differentially private.

(3) The algorithm uses $\mathcal{O}\left(\frac{\sqrt{W}\log^4 W}{\varepsilon\alpha}\right)$ bits of space and operations per update.

Proof. Consider Algorithm 7. For each $\ell \in [L]$, the stream is partitioned into blocks of length $S_\ell := 2^{\ell-2} \cdot \frac{\alpha\sqrt{W}}{100\log W}$ and then MisraGries is run on each block of length S_ℓ with threshold $\frac{\varphi\alpha\sqrt{W}}{16 \cdot S_\ell \log^3 W}$, with $\varphi = \min(\varepsilon, 1)$. By Fact 4.2, the sensitivity of each block is $\frac{\varphi\alpha\sqrt{W}}{16\log^3 W}$. Thus by Theorem 2.5, it suffices to add Laplacian noise $\mathsf{Lap}\left(\frac{\sqrt{\alpha W}}{8\log^2 W}\right)$ to each block to achieve $\left(\frac{\varepsilon}{2\log W}, 0\right)$-differential privacy. Then by composition, i.e., Theorem 2.9, we obtain $(\varepsilon, 0)$-differential privacy.

Moreover, we have that the sum of $\mathcal{O}(\log W)$ random variables drawn from the distribution $\mathsf{Lap}\left(\frac{\alpha\sqrt{W}}{8\log^2 W}\right)$ is at most $\frac{\alpha\sqrt{W}}{4}$ with high probability. Thus by Lemma A.2, we have $|f_k - \widehat{f_k}| \le \frac{\alpha\sqrt{W}}{2}$ for each $k \in \mathcal{L}$ with high probability.

By Theorem 4.1, running MisraGries on each block of size S_ℓ uses

$$\mathcal{O}\left(\frac{S_\ell \cdot \log^3 W}{\varepsilon\alpha\sqrt{W}}\right) = \mathcal{O}\left(\frac{2^\ell \cdot \log^2 W}{\varepsilon}\right)$$

bits of space and update time per operation. There are at most $\frac{W}{S_\ell} = \mathcal{O}\left(\frac{\sqrt{W}\log W}{2^\ell \alpha}\right)$ blocks of size S_ℓ, thus the total space and update time per operation across the blocks of size S_ℓ is $\mathcal{O}\left(\frac{\sqrt{W}\log^3 W}{\varepsilon\alpha}\right)$. Finally, we have $\ell \in [L]$, where $L = \mathcal{O}(\log W)$, so the total space and update time per operation of Algorithm 7 is $\mathcal{O}\left(\frac{\sqrt{W}\log^4 W}{\varepsilon\alpha}\right)$. □

Finally, we remark that for a window of W updates, we have $L_2(t-W+1:t) \geq \sqrt{W}$. Thus Theorem A.3 outputs a list $\mathcal{L}$ such that $k \in \mathcal{L}$ for each $k \in [n]$ with $f_k \geq \alpha\sqrt{W}$ and in particular, $f_k \geq \alpha\, L_2(t-W+1:t)$. Moreover, the algorithm achieves additive error $\frac{\alpha\sqrt{W}}{2} \leq \frac{\alpha\, L_2(t-W+1:t)}{2}$ for each $k \in \mathcal{L}$. Therefore, Theorem A.3 not only improves upon the continual L_1-heavy hitters of [Upa19], but it also solves the continual L_2-heavy hitters problem.