Private Decayed Predicate Sums on Streams

Jean Bolot
Technicolor
jean.bolot@technicolor.com

Nadia Fawaz
Technicolor
nadia.fawaz@technicolor.com

S. Muthukrishnan
Rutgers University
muthu@cs.rutgers.edu

Aleksandar Nikolov
Rutgers University
anikolov@cs.rutgers.edu

Nina Taft
Technicolor
nina.taft@technicolor.com

ABSTRACT

In many monitoring applications, recent data is more important than distant data. How does this affect privacy of data analysis? We study a general class of data analyses — predicate sums — in this context.

Formally, we study the problem of estimating predicate sums *privately*, for sliding windows and other decay models. While we require accuracy in analysis with respect to the decayed sums, we still want differential privacy for the entire past. This is challenging because window sums are not monotonic or even near-monotonic as the problems studied previously [DPNR10].

We present accurate ε-differentially private algorithms for decayed sums. For window and exponential decay sums, our algorithms are accurate up to additive $1/\varepsilon$ and polylog terms in the *range* of the computed function; for polynomial decay sums which are technically more challenging because partial solutions do not compose easily, our algorithms incur additional relative error. Our algorithm for polynomial decay sums generalizes to arbitrary decay sum functions. The algorithm crucially relies on our solution for the window sum problem as a subroutine. Further, we show lower bounds, tight within polylog factors and tight with respect to the dependence on the probability of error. Our results are obtained via a natural dyadic tree we maintain, but the crux is we treat the tree data structure in non-uniform manner.

We also extend our study and consider the "dual" question of maintaining conventional running sums on the entire data thus far, but when privacy constraints expire with time. We define a new model of privacy with expiration and consider the problems of designing accurate running sum and linear map algorithms in this model. Now the goal is to design algorithms whose accuracy guarantees scale with the *size of the privacy window*. We reduce running sum with a privacy window W to window sum without privacy expiration,and characterize the accuracy of output perturbation for general linear maps with privacy window W.

EDBT/ICDT '13, March 18 – 22 2013, Genoa, Italy

Categories and Subject Descriptors

K.4.1 [**Computers and Society**]: Privacy; H.2.8 [**Database Applications**]: Statistical Databases

General Terms

Algorithms, Security, Theory

Keywords

Differential Privacy, Continual Privacy, Decayed Sums, Online Algorithms

1. INTRODUCTION

Any nontrivial physical, hardware or software system has a dashboard continually observing the system variables, and updating various measurements. In such applications, data arrives over time, and we need to continually output the result of some analysis f for each time instant j on all data seen thus far. This challenges privacy of analysis because the same function is computed on several deltas of the data and the collection of these function values can potentially leak information. Recently, the notion of differential privacy was adopted to address this challenge [DPNR10, CSS10a], and we extend that study.

[DPNR10, CSS10a] identified the problem of computing the running sum of a series of 0/1 updates as an important technical primitive, formulated differential privacy of computing these running sums, and presented upper and lower bounds on accuracy of ε-differentially private algorithms for computing running sums. They showed that an additive accuracy of $O(\frac{1}{\varepsilon}\log^2 T)$ for all running sum values with constant probability is possible, and that $\Omega(\log T)$ additive error was necessary to answer privately all running sum queries for all time steps $j \in [1, T]$.

Power of Running Sums. The sums problem captures many analyses by applying a suitable predicate to the data items that map them to 0/1. For example, at time j, say data item $D_j = (u_j, m_j)$ is the user ID u_j and the name of the movie m_j watched in an online service by u_j at that time. A natural predicate is $\mathcal{P}_m(D_j) = 1$ if $m_j = m$ and 0 otherwise; the running sum with this predicate counts the number of user IDs that watched a particular film m. Another natural predicate is $\mathcal{P}_u(D_j) = 1$ if $u_j = u$ and 0 otherwise; this running sum counts the number of movies watched by a user u. The predicates can be different for different items. E.g., $\mathcal{P}_{j,u}(D_j) = 1$ if $u_j = u$ and $j \in [9, 17]$ will filter movies watched by user u during business hours 9 AM to 5 PM.

Even more generally, $\mathcal{P}$ may be a machine learning based classification routine such as whether a click by any user from a certain IP address on an Internet ad is a spam or not, and the running sum will count the total number of spam clicks from the given IP address. ■

Our point of departure from prior work is two fold.

- (*Primal: Decayed sums*) In reality, monitoring applications emphasize recent data more than data long past, as recent events are more relevant to the statistics being monitored. For example, monitoring applications typically consider a "window"' of continual observations such as, last T time units, or last W updates. More generally, they discount items based on how far they are in the past, and analyze decayed data. The commonly useful decay models are exponential and polynomial decays [DGIM02, CS03]. We study continual privacy of such decayed sum analyses. Notice that privacy guarantees need to be with respect to the entire data of the past, but accuracy needs to be with respect to the window, or decayed sum.

- (*Dual: Decayed privacy*) We address the dual problem when privacy concerns for events in the distant past get relaxed. We present an extension of the definition of differential privacy under continual observation that models situations in which knowledge of whether an event occurred is considered sensitive information *only* for relatively current events. The relaxation of privacy guarantees can be sharp — no privacy constraints are put on events that occurred more than W time steps in the past — or smooth, similarly to the data decay models above. We term our extension of differential privacy "privacy with expiration".

Our results are as follows.

Decayed Sum Results. At each time step i the algorithm receives a bit x_i; at each time step j, the algorithm is required to report an approximation $\hat{F}(x_1, \ldots, x_j)$ to a function $F(x_1, \ldots, x_j)$, and the entire sequence of values output thus far needs to be ε-differentially private. We use the notion of (δ, γ)-utility, satisfied by algorithms that at any time step j output a value $\hat{F}(x_1, \ldots, x_j)$ which is within δ absolute error from $F(x_1, \ldots, x_j)$ with probability $1 - \gamma$. Below we summarize our results for sufficiently small γ (results for larger γ can be found in the body of the paper):

- (*Window Sum*) The *window sum* problem with window size W requires estimating $F_w(j, W) = \sum_{i=j-W+1}^{j} x_i$ for each j. Further, the whole sequence F_w of outputs, for all j, should be ε-differentially private.

  We present an algorithm that achieves (δ, γ)-utility for $\delta = O(\frac{1}{\varepsilon} \log W \log \frac{1}{\gamma})$ (in the regime $\log W \leq \log \frac{1}{\gamma}$). While a window sum can be reduced to computing the difference of two running sums, existing running sum algorithms [DPNR10, CSS10a] achieve error $\delta = \Theta(\frac{1}{\varepsilon} \log T \log \frac{1}{\gamma})$, which can be much larger than the range W of F_w, and therefore, as bad as the trivial algorithm that outputs a fixed value independently of the input.

  We also present a lower bound of $\Omega(\min\{W/2, \frac{1}{\varepsilon} \log \frac{1}{\gamma}\})$. Note that the dependence on the error probability γ is optimal. This lower bound generalizes a previous lower for the running sum problem [DPNR10].

- (*Exponential Decay*) The *exponential decay sum* problem is to estimate $F_e(j, \alpha) = \sum_{i=1}^{j} x_i \alpha^{j-i}$ accurately, while the whole sequence F_e of outputs, for all j, should be ε-differentially private.

  We present an algorithm that achieves (δ, γ)-utility with $\delta = O(\frac{1}{\varepsilon} \log \frac{\alpha}{1-\alpha} \log \frac{1}{\gamma})$. We also present a lower bound of $\Omega\left(\min\left\{\frac{\alpha}{1-\alpha}, \frac{\log(1/\gamma)}{\varepsilon}\right\}\right)$. Once again, the dependence on the error probability γ is optimal. Unlike F_w, F_e at each time step depends on the entire sequence of updates; nevertheless, our algorithm achieves bounded error, polylogarithmic in the range of F_e.

- (*Polynomial Decay and Other Decay Functions*) The *polynomial decay sum* problem is to estimate $F_p(j, c) = \sum_{i=1}^{j} \frac{x_i}{(j-i+1)^c}$ accurately, while the whole sequence F_p of outputs, for all j, should be ε-differentially private. In general, one can also consider other decay sum functions $\sum_{i=1}^{j} x_i g(j-i)$.

  We present an algorithm that for each j returns $(1 \pm \beta) F_p(j, c) \pm \left(\frac{1}{c\beta^2} \log \frac{1}{1-\beta}\right) \log \frac{1}{\gamma}$ with probability $1 - \gamma$. We also present a lower bound of $\Omega\left(1 - \frac{\varepsilon^{c-1}}{\log^{c-1}(1/\gamma)}\right)$ against *purely* additive error. The polynomial decay problem presents a greater challenge than window sums or exponential decay since there is no direct way to combine a polynomial decay sum over an interval $[a, b]$ and $[b, c]$ into a polynomial decay sum over $[a, c]$. We develop a *general technique* that works on a large class of decay sum functions (including polynomial decay) and reduces the problem of estimating the decay sum to keeping multiple window sums in parallel. The technique results in a bi-criteria approximation, because of which our lower and upper bounds are incomparable for this problem.

In comparison with the simple randomized response strategy [War65] (i.e. with probability $1/2 - \varepsilon/2$ change update x_i to $1 - x_i$ and keep exact statistics of the changed input), our algorithms achieve exponentially smaller additive error: randomized response leads to estimators with standard deviation proportional to the "energy" of the decay function, while our estimators have standard deviation polylogarithmic in the energy. Technically,

- our algorithms keep dyadic tree data structures as is natural and also used in [DPNR10, CSS10a] and elsewhere. However, in order to provide estimates with error polylogarithmic in the range of the decay function, we need to treat the dyadic tree data structure in non-uniform manner: either adding different noise at different nodes, or weighing the contribution of an update to different nodes differently, which is our technical contribution;
- we derive all our lower bounds from a common framework, based on standard packing arguments. We extend prior work in two ways: the lower bounds apply to decay sum problems that have not been considered before, and they apply against the weaker (δ, γ)-utility guarantee (rather than requiring that all queries are accurate, as in [DPNR10]).

Privacy with Expiration. Intuitively, differential privacy guarantees that the output of an algorithm is almost unchanged in distribution no matter if any sensitive event took place or not. In the case of privacy under continual observation, we insist that this guarantee holds at each time step for all outputs produced up to that time step. To extend differential privacy so that we can model expiring privacy concerns, we require that the output of the monitoring algorithm is almost unchanged in distribution no matter if any *recent* sensitive event took place or not. We model the notion of recent by imposing a sliding window on the privacy guarantees of events.[1]

Trivially, any ε-differentially private algorithm also satisfies the requirements of privacy with expiration. However, it is an interesting question whether the relaxed privacy guarantees allow for higher accuracy. Older updates which are no longer considered sensitive may be used "in the clear"; however, as the newer updates must maintain their privacy continuously, it is not immediately obvious how to use the older updates to improve accuracy. Since the problems of maintaining decayed sums and maintaining a running sum under decaying privacy constraints are morally dual to each other, it is natural to ask whether they have the same error complexity. We provide a positive answer when privacy concerns expire according to a sharp threshold: the running sum problem under differential privacy with expiration window W can be reduced to the problem of computing window sums with window size W under differential privacy without expiration. The reduction preserves accuracy exactly. Therefore, using our result for window sums, we get an algorithm for running sums with privacy expiration window W which adds error polylogarithmic in W rather than the size of the full input.

Another natural question is how the error complexity of general classes of functions changes when we expire privacy. In addition to running sums, we consider the general problem of computing linear transformations of the input online. Formally, at each time step the algorithm is required to approximate an inner product $\langle \mathbf{a}_i, \mathbf{x}^i \rangle$, where $\mathbf{x}^i$ is the input up to time i. Again, we require privacy only with respect to the last W time updates. We characterize the error achievable by output perturbation under privacy with expiration. Once again, the error scales with W rather than the full input size, showing that expiring privacy constraints allows for better accuracy guarantees.

A summary of accuracy guarantees for running sum when either the relevance of updates or privacy concerns are decayed with time is presented in Table 1.

2. DISCUSSION OF PRIOR WORK

The problem of tracking statistics on dynamic data while preserving privacy under continual observation is introduced in [DPNR10] (a preliminary version was presented in an invited talk by Dwork [Dwo10]). In [DPNR10], an algorithm private under continual observation is presented for the running sum problem. For any fixed time step, their algorithm achieves additive error of $O(\frac{1}{\varepsilon}\log^{1.5} T)$ with constant probability, where T is an upper bound on the maximum size of the input, known to the algorithm. Independently, [CSS10a] presented a continually private algorithm for the running sum problem that at any step j, guarantees an additive error of $O(\frac{1}{\varepsilon}\log^{1.5} j)$ with constant probability. This matches [DPNR10], while not knowing T.

The algorithm of [CSS10a] is related to our work: our algorithm for window sum reduces to their algorithm for running sum when the size of the window coincides with the size of the input. However, if their algorithm or the algorithm in [DPNR10, CSS10b] is used directly to compute window sums, then the error at time j will be on the order of $\log^{1.5} j$ and for large j will overcome the window size. Further, algorithms in [CSS10a, DPNR10, CSS10b] do not work for decayed sums.

[DPNR10] shows how to transform a private *streaming* algorithm that satisfies a monotonicity property to a private, continual algorithm. However, accurate decayed sum estimators do not have the monotonicity property, and the general transformation cannot be used.

3. NOTATION AND PRELIMINARIES

Online Data Model We consider online problems with binary input: at each time step i the algorithm receives input $x_i \in [a, b]$; and is required to report an approximation $\hat{F}(x_1, \ldots, x_i)$ to a function $F(x_1, \ldots, x_i)$. We present our upper bounds for $a = 0, b = 1$. For general a, b, our absolute error bounds scale linearly in $b - a$.

Decayed Sum Problems The functions F we are interested in approximating are *decayed sum* functions. Consider a non-increasing function $g : \mathbb{N} \to \mathbb{R}^+$ such that $g(0) = 1$. The *decayed sum induced by* g is the function $F(j) = F(x_1, \ldots, x_j) = \sum_{i=1}^{j} x_i g(j - i)$. I.e., F is the convolution of the input and a non-increasing function g. The decayed sum problems we consider are defined below

- when $g(i) = 1\ \forall i$, the *running sum* problem (considered in [CSS10a, Dwo10, DPNR10]): $F_s(j) = \sum_{i=1}^{j} x_i$.
- when $g(i) = \mathbf{1}_{\{i<W\}}$, the *window sum* problem (with window size W): $F_w(j, W) = \sum_{i=j-W+1}^{j} x_i$. To simplify notation, in the above definition we assume that $x_i = 0$ for all $i \leq 0$.
- when $g(i) = \alpha^i$ $(\alpha < 1)$, the *exponential decay sum* problem: $F_e(j, \alpha) = \sum_{i=1}^{j} x_i \alpha^{j-i}$.
- when $g(i) = (i + 1)^{-c}$ $(c > 1)$, the *polynomial decay sum* problem: $F_p(j, c) = \sum_{i=1}^{j} \frac{x_i}{(j-i+1)^c}$.

The last three problems have not been considered in the differential privacy literature before, and specifically not in the continual observation model. The problems of keeping event counts and other statistics over windows [DGIM02] and keeping decayed (in particular exponential and polynomial decay) sums [CS03] have been studied in the field of small space streaming algorithms.

Differential Privacy We use the standard definition of differential privacy, applied to the online data model:

DEFINITION 1 ([DMNS06, DPNR10]). *Let $\mathcal{A}$ be a randomized online algorithm that at time step j outputs the real value $\hat{F}(x_1, \ldots, x_j)$. $\mathcal{A}$ satisfies* ε-differential privacy *if for all $T \in \mathbb{Z}$, for all measurable subsets $S \subseteq \mathbb{R}^T$, all possible*

[1] One may also relax privacy guarantees of old events in a smooth way, and our definition allows for such modifications.

	Running Sum	Decaying Relevance	Expiring Privacy
ε-**Differential Privacy**	whole input $\mathbf{x}$	whole input $\mathbf{x}$	sliding window W_p of ε
(δ,γ)-**Accuracy**	whole input $\mathbf{x}$	sliding window W of $\mathbf{x}$	whole input $\mathbf{x}$
UB δ Output Pert.	$\Theta(\frac{1}{\varepsilon}\frac{T}{\sqrt{\gamma}})$	$\Theta(\frac{1}{\varepsilon}\frac{W}{\sqrt{\gamma}})$	$O(\frac{1}{\varepsilon}W_p \log\frac{1}{\gamma})$
UB δ Input Pert.	$\Theta(\frac{1}{\varepsilon}\sqrt{T\log\frac{1}{\gamma}})$	$\Theta(\frac{1}{\varepsilon}\sqrt{W\log\frac{1}{\gamma}})$	$O(\frac{1}{\varepsilon}\sqrt{W_p\log\frac{1}{\gamma}})$
UB δ Diadic Tree	$\Theta(\frac{1}{\varepsilon}\log^{1.5} T\log\frac{1}{\gamma})$	$O(\frac{1}{\varepsilon}\log^{1.5} W\log\frac{1}{\gamma})$	$O(\frac{1}{\varepsilon}\log^{1.5} W\log\frac{1}{\gamma})$
LB δ	$\Omega(\log T\log\frac{1}{\gamma})$	$\Omega(\min\{W/2, \frac{1}{\varepsilon}\log\frac{1}{\gamma}\})$	

Table 1: Comparing accuracy guarantees for running sums when data relevance or privacy concerns decay with time.

inputs $x_1,\ldots,x_T$, all j and all $x'_j \in [0,1]$ (where probability is over the coin throws of $\mathcal{A}$)

$$\Pr[(\hat{F}(x_1,\ldots,x_j,\ldots,x_k))_{k=1}^T \in S] \leq e^{\varepsilon}\Pr[(\hat{F}(x_1,\ldots,x'_j,\ldots,x_k))_{k=1}^T \in S].$$

This is the standard definition of differential privacy as proposed by Dwork et al. [DMNS06], but with the modification that the algorithm receives the input online and produces output at every step, and the whole sequence of outputs is available to an adversary. This model of privacy for online algorithms operating on time series data, termed *privacy under continual observation*, was introduced by [DPNR10], with preliminary results presented in [Dwo10].

We use the following basic facts about differential privacy. The first theorem gives a simple way to achieve differential privacy for algorithms with numerical output, based on adding random noise scaled according to the sensitivity of the statistic being computed. The second fact is that composing multiple privacy mechanisms results in smooth privacy loss.

THEOREM 1 ([DMNS06]). *For a function $F : [a,b]^T \to \mathbb{R}^d$, let the* sensitivity *of F, S_F be the smallest real number that satisfies $\forall x_1,\ldots,x_T, \forall j \in [T], \forall x'_j \in [a,b]$:*

$$\|F(x_1,\ldots,x_j,\ldots,x_T) - F(x_1,\ldots,x'_j,\ldots,x_T)\|_1 \leq S_F$$

Then an algorithm that outputs $\hat{F}(x_1,\ldots,x_T) = F(x_1,\ldots,x_T) + \mathrm{Lap}(S_F/\varepsilon)^d$ satisfies ε-differential privacy, where $\mathrm{Lap}(\lambda)^d$ is a sample of d independent Laplace random variables with mean 0 and scale parameter λ.

THEOREM 2 ([DMNS06]). *Let algorithm $\mathcal{A}_1$ satisfy ε_1-differential privacy and algorithm $\mathcal{A}_2$ satisfy ε_2-differential privacy. Then an algorithm $\mathcal{A}$ that on input $\mathbf{x} = \{x_1,\ldots,x_T\}$ outputs $\mathcal{A}(\mathcal{A}_1(\mathbf{x}),\mathcal{A}_2(\mathbf{x}))$ satisfies $(\varepsilon_1+\varepsilon_2)$-differential privacy.*

Utility We adopt the following, commonly used notion of utility:

DEFINITION 2. *Let $\mathcal{A}$ be a randomized online algorithm that at time step j outputs $\hat{F}(x_1,\ldots,x_j) \in \mathbb{R}$. Then, $\mathcal{A}$ achieves (δ,γ)-utility with respect to a function F, if for all j, $\Pr[|\hat{F}(x_1,\ldots,x_j) - F(x_1,\ldots,x_j)| > \delta] < \gamma$.*

Dyadic Tree Datastructure We repeatedly use the following *dyadic tree* data structure which is common in algorithmics. This data structure is a balanced augmented search tree and variants of it are common in much algorithmic work.

Let $\mathcal{T}(L,U)$ be a complete binary tree of height $h = \log(U-L+1)+1$ (assuming, for simplicity, that $U-L+1$ is a power of 2). The leaves of the tree are indexed by the integers $L, L+1, \ldots, U$, and if two sibling nodes are indexed by the intervals $[l_1,u_1]$ and $[l_2 = u_1+1, u_2]$, then their parent is indexed by $[l_1,u_2]$. Note that at level k of the tree (the leaves being at level 1), the indexing intervals have the form $[L+(i-1)2^{k-1}, L+i2^{k-1}-1]$ for $i \in [1, 2^{h-k}]$. We call a node whose indexing interval precedes its sibling's indexing interval a *left node*; the sibling of a *left node* is a *right node*. With each node we associate a variable: for the node indexed by $[l,u]$, the associated variable is denoted c_{lu}. Given a tree $\mathcal{T} = \mathcal{T}(L,U)$ and a prefix interval $[L,u]$, we define function $s(u,\mathcal{T})$ recursively:

- if $[L,u]$ indexes a node in $\mathcal{T}$, then $s(u,\mathcal{T}) = c_{Lu}$;
- otherwise, let u' be the largest integer less than u such that $[L,u']$ indexes some node in $\mathcal{T}$; equivalently, u' is the largest integer less than u that can be written as $u' = L + 2^{k-1} - 1$. Let $\mathcal{T}'$ be the subtree of $\mathcal{T}$ rooted at the sibling of $[L,u']$ (indexed by $[u'+1, u'+2^{k-1}]$); then $s(u,\mathcal{T}) = c_{Lu'} + s(u,\mathcal{T}')$.

The following lemma is essential to our analysis and can be easily proved by induction.

LEMMA 1. *There exist $r \leq \lceil\log(u-L+1)\rceil$ integers $u_1, \ldots, u_r = u$ such that $s(u,\mathcal{T}) = c_{Lu_1} + \sum_{k=1}^{r-1} c_{u_k+1,u_{k+1}}$. Furthermore, all nodes indexed by $[u_k+1, u_{k+1}]$ are left nodes in $\mathcal{T}$, and each node is in a different level of $\mathcal{T}$.*

PROOF. The integers $u_1,\ldots,u_r$ are given directly by the recursive definition of $s(u,\mathcal{T})$. To bound r, consider that at each step in the recursion, unless $[L,u]$ indexes a node in $\mathcal{T}$, the tree $\mathcal{T}'$ has at most half the number of leaves of the smallest subtree of $\mathcal{T}$ that contains u as a leaf. Initially the smallest subtree that contains u as a leaf has number of leaves equal to the smallest power of 2 greater than or equal to $u-L+1$, i.e. the number of leaves initially is $2^{\lceil\log(u-L+1)\rceil}$. The recursion stops when we reach a tree with only a single node, and, therefore, we make at most $\lceil\log(u-L+1)\rceil$ recursive calls. The bound on r follows.

The condition that all nodes are left siblings follows from the fact each node is indexed by an interval that contains the leftmost leaf of the current subtree.

Finally, notice that the only way to pick two nodes on the same level is if after picking u', in the next step of the recursion we pick the root of $\mathcal{T}'$. However, in this case we would have picked the parent of $[L,u']$ instead of $[L,u']$, a contradiction. □

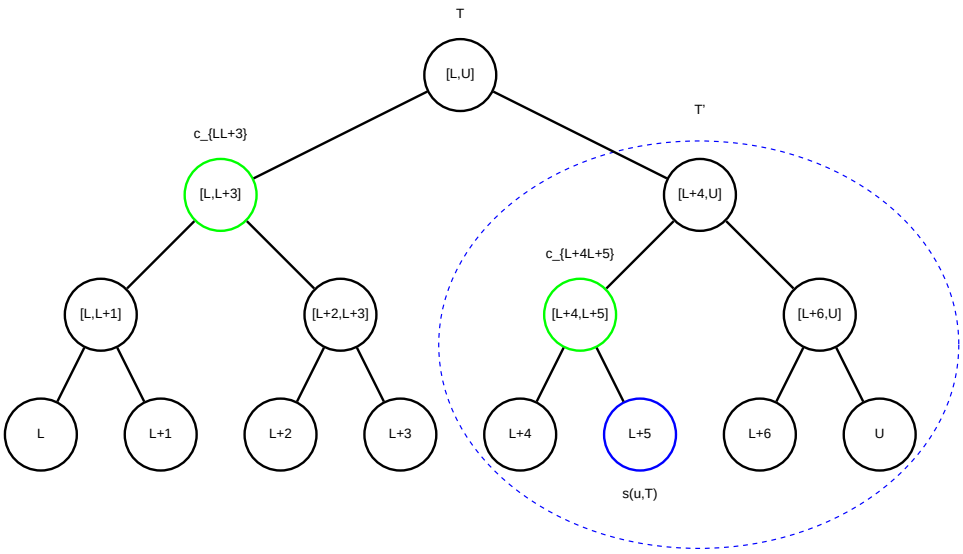

Figure 1: Dyadic tree data structure. In this example, $u = L + 5$ is shown in a blue node, $u' = L + 3$, and the prefix sum $s(u, \mathcal{T}) = c_{L,L+3} + c_{L+4,L+5}$ is obtained by adding the counters at the two green nodes $[L, L + 3]$ and $[L + 4, L + 5]$.

Chernoff Bound for Laplace Variables We will use the following Chernoff bound for sums of independent Laplace random variables.

LEMMA 2. *Let $s_1, \ldots, s_n$ be independent Laplace random variables such that $s_i \sim \text{Lap}(b_i)$. Denote $S = \sum_{i=1}^n s_i$ and $\sigma = \sqrt{2\sum_{i=1}^n b_i^2}$. Then, for all $\lambda < \min_i \frac{0.75}{b_i}$, we have* $\Pr[|S| \geq t\sigma] \leq 2\exp(0.75\lambda^2\sigma^2 - \lambda t\sigma)$.

PROOF. We use the standard technique of bounding the moment generating function of S and applying Markov's inequality. Details follow.

Since the distribution of S is symmetric, we have $\Pr[|S| \geq t\sigma] = 2\Pr[S \geq t\sigma]$. For any λ, we have:

$$\Pr[S \geq t\sigma] = \Pr[e^{\lambda S} \geq e^{\lambda t\sigma}] \leq \frac{\mathrm{E}[e^{\lambda S}]}{e^{\lambda t\sigma}} = \frac{\prod_{i=1}^n \mathrm{E}[e^{\lambda s_i}]}{e^{\lambda t\sigma}} \quad (1)$$

For $\lambda < 1/b_i$, the moment generating function of the Laplace random variable s_i is $\mathrm{E}[e^{\lambda s_i}] = 1/(1 - \lambda^2 b_i^2)$. Assuming $\lambda b_i \leq .75$, we have

$$\mathrm{E}[e^{\lambda s_i}] = \frac{1}{1 - \lambda^2 b_i^2} < \exp(\frac{3}{2}\lambda^2 b_i^2).$$

Substituting into (1), we get

$$\Pr[S \geq t\sigma] \leq \exp(\frac{3}{2}\lambda^2 \sum_{i=1}^n b_i^2 - \lambda t\sigma),$$

as desired. □

4. UPPER BOUNDS

4.1 Window Sum

A key observation for computing window sums with error polylogarithmically bounded in W is that, unlike with running sum, only the lowest $\log W + 1$ layers of the dyadic tree are necessary to compute window sum. However, if we keep a dyadic tree for every window of size W, each update will contribute to more than W variables, resulting in data structures with large sensitivity, which, for differential privacy, translates into more noise. Our main idea is that instead of keeping a dyadic tree for every window, we can divide the input into blocks of size W, and view the windows that span two blocks as the union of a suffix and a prefix of two blocks.

Algorithm 1 WINDOWSSUM

For $k \geq 1$, define $\mathcal{T}_k = \mathcal{T}((k-1)W + 1, kW)$, with all c_{lu} initialized to $\text{Lap}((\log W + 1)/\varepsilon)$.
for all inputs x_i **do**
  add x_i to all c_{lu} in $\mathcal{T}_{\lceil i/W \rceil}$ such that $i \in [l, u]$.
  output: $\hat{F}_w(i, W) = s((k-1)W, \mathcal{T}_{k-1}) - s(i-W, \mathcal{T}_{k-1}) + s(i, \mathcal{T}_k)$, where $k = \lceil i/W \rceil$.
end for

The algorithm WINDOWSUM is shown as Algorithm 1. In the remainder of this section we assume that W is an exact power of 2.

THEOREM 3. WINDOWSUM *satisfies ε-differential privacy, and achieves (δ, γ)-utility with*

$$\delta = \begin{cases} O(\frac{1}{\varepsilon}\log^{1.5} W \log^{0.5}\frac{1}{\gamma}), & \log W \geq \log\frac{1}{\gamma} \\ O(\frac{1}{\varepsilon}\log W \log\frac{1}{\gamma}), & \log W < \log\frac{1}{\gamma} \end{cases}$$

Furthermore, WINDOWSUM *can be implemented to use $O(W)$ words of space and to run in $O(\log W)$ time per update.*

PROOF. **Privacy.** Observe that any variable c_{lu} used to compute $\hat{F}_w(j, W)$ satisfies $l \leq u \leq j$. Therefore, the counters c_{lu} that contribute to $\hat{F}_w(j, W)$ will not be updated after time step j and $\hat{F}_w(j, W)$ will be identically distributed if it is computed at any time step $T \geq j$, so for the analysis we can assume that all outputs are produced at time step T.

Next we fix T and argue that WindowSum is ε-differentially private for inputs of size T. Since the choice of T is arbitrary, privacy for all T follows. For this purpose, let $\mathbf{c}(\mathbf{x})$ be the vector of the values of all variables (in an arbitrary order) c_{lu} such that $u \le T$ when the input is $\mathbf{x} = (x_1, \ldots, x_T)$. Let also $\mathbf{c}_0(\mathbf{x})$ be $\mathbf{c}(\mathbf{x})$ with the initializing Laplace noise removed. Since each x_j contributes to exactly $\log W + 1$ variables c_{lu}

$$\forall j \in [T], \forall x'_j \in [0,1]: \\ \|\mathbf{c}_0(x_1, \ldots, x_j, \ldots, x_T) - \mathbf{c}_0(x_1, \ldots, \quad x'_j, \ldots, x_T)\|_1 \\ \le \log W + 1.$$

Differential privacy of $\mathbf{c}(\mathbf{x})$ follows from above and Theorem 1. Since the sequence of outputs of WindowSum up to time step T is a deterministic function of $\mathbf{c}(\mathbf{x})$, privacy of WindowSum follows.

Accuracy. It is easy to see that $\mathrm{E}\hat{F}_w(j,W) = F_w(j,W)$. By Lemma 1, for each k and each u, $s(u, \mathcal{T}_k)$ is the sum of at most $\log W$ random variables, each with variance $2(\log W + 1)^2/\varepsilon^2$. Therefore, the standard deviation σ of $\hat{F}_w(j,W)$ is $O(\log^{1.5} W/\epsilon)$.

We consider two cases. For the first case, let $t = 2\sqrt{\ln(1/\gamma)}$ and $\lambda = \phi\frac{t}{\sigma}$ for a constant ϕ to be determined later. By Lemma 2, as long as $\lambda < 0.75\varepsilon/(\log W + 1)$, we have that $\Pr[|\hat{F}_w(j,W) - F_w(j,W)| > C\sqrt{\log(1/\gamma)}\sigma] < \gamma$ for some fixed constant C that depends on ϕ. A calculation shows that as long as $\log W \ge \log(1/\gamma)$, the minimum value of ϕ such that the constraint on λ holds can be bounded below by a constant. This completes the analysis of the first case.

For the second case, when $\log W < \log(1/\gamma)$, we set the following parameters: $\eta = \log_{\ln(1/\gamma)} \ln W$ (notice that $\eta < 1$); $t = C'\frac{\ln(1/\gamma)}{\sqrt{\ln W}}$, and $\lambda = \frac{t^{\eta/(2-\eta)}}{\sigma} = \frac{C'^{\eta/(2-\eta)}\sqrt{\ln W}}{\sigma}$, where C' is a constant chosen so that $\lambda < 0.75\varepsilon/(\log W + 1)$ holds. Applying Lemma 2, we have that for a value C that depends on C', $\Pr[S \ge C\frac{1}{\varepsilon}\log(1/\gamma)\log W] \le \exp(\Omega(t^{2/(2-\eta)})) = \exp(-\Omega(\ln 1/\gamma))$.

For the running time and space complexity analysis, notice that each update requires accessing $O(\log W)$ nodes, and that only the last two dyadic trees need to be stored. □

We can also show that we can approximate window sums *simultaneously* for all window sizes and preserve privacy under continual observation. Our approximation is different for different window sizes W, and for any particular W, it is almost the same as that of Theorem 3. Details can be found in Appendix 4.2.

4.2 Window Sum Simultaneously for all W

Here we give an algorithm that works simultaneously for all window sizes. Our main observation is that if for window size W we divide the input into blocks of size $W' \in [W, 2W]$ instead of exactly W as in WindowSum, then we can store all necessary dyadic tree data structures as subtrees of a single dyadic tree. However, storing the whole dyadic tree with the same noise at any level will result in error of size $\Omega(\log^{1.5} T)$ for all W. Instead, we want to make sure that within a subtree of height h, the noise added to any variable is proportional to h. To achieve this, we use a different privacy parameter ε_k at level k of the dyadic tree and ensure that the sum of privacy parameters converges to ε.

Let $\beta > 1$ be a parameter and $\zeta(\cdot)$ be the Riemann zeta function: $\zeta(\beta) = \sum_1^\infty i^{-\beta}$. Set $\epsilon_k = \frac{\varepsilon}{\zeta(\beta)k^\beta}$. The algorithm

Algorithm 2 AllWindowSum

Initialize $\mathcal{T} = \mathcal{T}(1,1)$, with $c_{1,1}$ initialized to $\mathrm{Lap}(1/\varepsilon_1)$.
for all updates x_i **do**
  if the rightmost leaf of $\mathcal{T}$ is $i-1$ **then**
    Grow $\mathcal{T}$ so that $\mathcal{T} = \mathcal{T}(1, 2(i-1))$, adding additional nodes and variables as necessary; initialize new variables at level k to $\mathrm{Lap}(1/\varepsilon_k)$.
    Add the value $c^0_{1,i-1}$ to the root variable $c_{1,2(i-1)}$, where c^0_{lu} is the value of c_{lu} without the Laplace noise.
  end if
  Add x_i to all c_{lu} in such that $i \in [l,u]$.
  Let $W' = 2^{\lceil \log W \rceil}$. At time step j, output $\hat{F'}_w(j,W) = s((k-1)W, \mathcal{T}_{k-1}) - s(j-W, \mathcal{T}_{k-1}) + s(j, \mathcal{T}_k)$, where $k = \lceil j/W' \rceil$.
end for

AllWindowSum is shown as Algorithm 2. Proof of theorem below is analogous to Theorem 3.

Theorem 4. *There exists a constant K s.t.* AllWindowSum *satisfies ε-differential privacy and achieves (δ,γ)-utility, where*

$$\delta = \begin{cases} O(\frac{1}{\varepsilon}\log^{1.5\beta} W \log^{0.5}\frac{1}{\gamma}), & \log W \ge K\log\frac{1}{\gamma} \\ O(\frac{1}{\varepsilon}\log^{\beta} W \log\frac{1}{\gamma}), & \log W < K\log\frac{1}{\gamma} \end{cases}$$

Furthermore, the algorithm can be implemented to use $O(T)$ words of space and run in $O(\log T)$ time per update on inputs consisting of T updates.

Proof. **Privacy.** The proof of privacy is analogous to the proof of privacy for Theorem 3, but we treat different levels of $\mathcal{T}$ separately and use Theorem 2 to bound the total privacy loss. More precisely, we show that level k in the tree satisfies ε_k-differential privacy and use the fact that $\sum_{k=1}^\infty \varepsilon_k = \varepsilon$.

Utility. The utility analysis is also analogous to the proof of Theorem 3, noticing the following facts: **(1)** $W \le W' \le 2W$; **(2)** as an upper bound on the variance of any variable used to compute $\hat{F'}_w(j,W)$ we can use the variance of variables at level $\log W' + 1$, which is $O(\log^\beta W)$. The rest of the proof is unchanged. □

4.3 Exponential Decay

While for the window sum problem we keep a sequence of dyadic trees, for the exponential decay problem we keep a single dyadic tree that grows over time. The main property of exponentially decaying sums that we use is that if S_1 is the exponential decay sum over a time interval $[a, b-1]$ and S_2 is the exponential decay sum over a time interval $[b,c]$, then $\alpha^{c-b+1}S_1 + S_2$ is the exponential decay sum over the time interval $[a,c]$. Thus at a node in the dyadic tree that is indexed by interval $[l,u]$ we can keep the exponential decay sum for that interval. However, doing this for every interval results in a data structure with unbounded sensitivity. We update only the left nodes in the tree and show that we can bound the sensitivity in that case.

The ExponentialSum algorithm is shown as Algorithm 3. We analyze the algorithm for $\alpha \in (2/3, 1)$; observe that when $\alpha \le 2/3$, the range of the F_e is $[0,3]$, and, therefor, achieving $(1.5, 0)$-utility is trivial. Thus $\alpha \to 1$ is the interesting regime for approximating F_e.

The following lemma is useful in the analysis.

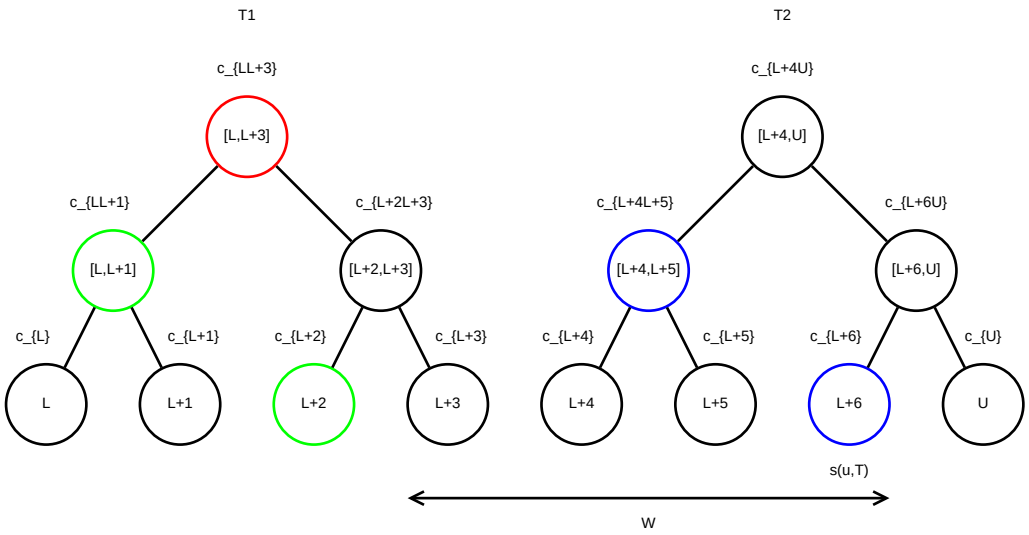

Figure 2: Window sum for a window size $W = 4$. This example illustrates the algorithm at time $i = 7$, and the output is $\hat{F}_w(7, W = 4) = s(W, \mathcal{T}_1) - s(3, \mathcal{T}_1) + s(7, \mathcal{T}_2) = c_{14} - (c_{12} + c_3) + (c_{56} + c_7) = x_4 + x_5 + x_6 + x_7 + z_{14} - z_{12} - z_3 + z_{56} + z_7$, where z_{lu} denotes the noise at node $[l, u]$.

Algorithm 3 ExponentialSum

Set $\lambda = \frac{1}{\alpha \ln 2}\left(\ln \frac{2\alpha}{1-1\alpha} + \frac{1}{2} + \ln 2\right)$.
Initialize $\mathcal{T} = \mathcal{T}(1, 1)$, with $c_{1,1}$ initialized to $\mathrm{Lap}(\lambda/\varepsilon)$
for all updates x_i **do**
  if the rightmost leaf of $\mathcal{T}$ is $i - 1$ **then**
    Grow $\mathcal{T}$ so that $\mathcal{T} = \mathcal{T}(1, 2(i-1))$, adding additional nodes and variables as necessary and initializing new variables to $\mathrm{Lap}(\lambda/\varepsilon)$.
    Add the value $\alpha^{i-1} c^0_{1,i-1}$ to the root variable $c_{1,2(i-1)}$, where c^0_{lu} is the value of c_{lu} without the Laplace noise.
  end if
  for all $[l, u]$ such $i \in [l, u]$ and the node indexed by $[l, u]$ is a left node **do**
    add $x_i \alpha^{u-i}$ to c_{lu}
  end for
  output $\hat{F}_e(j, \alpha) = \sum_{k=0}^{r} c_{u_k, u_{k+1}} \alpha^{j - u_{k+1}}$.
end for

Lemma 3. *For an arbitrary i, let $[l_1, u_1], [l_2, u_2], \ldots$ be the sequence of intervals such that $\forall k : i \in [l_k, u_k]$ and $[l_k, u_k]$ is a left node. Assume the intervals are ordered in ascending order of $u_k - l_k$. Then $u_k - i \geq 2^{k-1} - 1$.*

Proof. By induction. The base case is trivial, as from $i \in [l_1, u_1]$ follows $u_1 - i \geq 0$. For the inductive step, it suffices to show that $u_k - u_{k-1} \geq 2^{k-2}$. By the construction of $\mathcal{T}$, all nodes indexed by intervals $[l, u]$ such that $i \in [l, u]$ lie on the path from the leaf indexed by i to the root of $\mathcal{T}$. Therefore, all nodes indexed by $[l_k, u_k]$ for some k are ancestors of i, and, by the construction of $\mathcal{T}$ we have $u_k - l_k + 1 \geq 2^{k-1}$; in particular, $[l_k, u_k]$ is an ancestor of $[l_{k-1}, u_{k-1}]$ and $u_{k-1} - l_{k-1} + 1 \geq 2^{k-2}$. By assumption, all nodes indexed by $[l_k, u_k]$ are left nodes; let the right sibling of $[l_{k-1}, u_{k-1}]$ be the node indexed by $[l'_{k-1}, u'_{k-1}]$. By construction, $u'_{k-1} - l'_{k-1} = u_{k-1} - l_{k-1}$ and the parent of both nodes is indexed by $[l_{k-1}, u'_{k-1}]$. All ancestors of $[l_{k-1}, u_{k-1}]$ are indexed by intervals that contain $[l_{k-1}, u'_{k-1}]$ as a subinterval, and, therefore,

$$\begin{aligned} u_k &\geq u'_{k-1} = u_{k-1} + (u_{k-1} - l_{k-1} + 1) \\ &\geq u_{k-1} + 2^{k-2} \end{aligned}$$

This completes the inductive step. □

Theorem 5. *Assume $\alpha \in (2/3, 1)$ and let K be a universal constant.* ExponentialSum *satisfies ε-differential privacy and achieves (δ, γ)-utility with*

$$\delta = \begin{cases} O(\frac{1}{\varepsilon} \frac{\alpha}{1-\alpha} \log^{0.5} \frac{1}{\gamma}), & \log \frac{\alpha}{1-\alpha} \geq \log \frac{1}{\gamma} \\ O(\frac{1}{\varepsilon} \log \frac{\alpha}{1-\alpha} \log \frac{1}{\gamma}), & \log \frac{\alpha}{1-\alpha} < \log \frac{1}{\gamma} \end{cases}$$

Furthermore, ExponentialSum *can be implemented to use $O(\log T)$ words of space and to run in $O(\log T)$ time per update on inputs consisting of T updates.*

Proof. **Privacy.** It is sufficient to fix T and argue that ExponentialSum is ε-differentially private for inputs of size T when all outputs for $j \leq T$ are produced at step T.

We analyze the sensitivity of $\mathcal{T}$. Define $\mathbf{c}_0(\mathbf{x})$ as in the proof of Theorem 3 and $[l_1, u_1], [l_2, u_2], \ldots$ as in Lemma 3. We have

$$\begin{aligned} &\|\mathbf{c}_0(x_1, \ldots, x_i, \ldots, x_T) - \mathbf{c}_0(x_1, \ldots, 1 - x_i, \ldots, x_T)\|_1 \\ &\quad \leq \sum_{k=1}^{\infty} \alpha^{u_k - i} x_i \leq \sum_{k=1}^{\infty} \alpha^{u_k - i} = \frac{1}{\alpha} \sum_{k=0}^{\infty} \alpha^{2^k} \\ &\quad \leq \frac{1}{\alpha} + \frac{1}{\alpha} \int_0^{\infty} \alpha^{2^x} dx = \frac{1}{\alpha} + \frac{1}{\alpha \ln 2} \int_{\ln \frac{1}{\alpha}}^{\infty} \frac{e^{-t}}{t} dt \\ &\quad = \frac{\ln 2 + E_1(\ln \frac{1}{\alpha})}{\alpha \ln 2}. \end{aligned} \tag{2}$$

Here $E_1(x) = E_1(x) = \int_x^{\infty} \frac{e^{-t}}{t} dt$. We have the following series expansion for E_1, which converges for all real $|x| \leq$

π [AS64]:

$$E_1(x) = -\eta - \ln x + \sum_{k=1}^{\infty} \frac{(-1)^{k+1} x^k}{k!k}, \tag{3}$$

where η is the Euler-Mascheroni constant. Since, by assumption, $\alpha > e^{-1}$, we have $\ln \frac{1}{\alpha} < 1$. For $x < 1$, the last term in (3) is bounded by $\eta + E_1(1) = \eta + \frac{1}{2}$. Therefore, we have,

$$E_1(2 \ln \frac{1}{\alpha}) \leq -\ln \ln \frac{1}{\alpha} + \frac{1}{2} = \ln \frac{1}{\ln \frac{1}{\alpha}} + \frac{1}{2} \tag{4}$$

For $x \in (0, 2)$, we have the following series expansion for $\ln x$:

$$\ln x = x - 1 - \sum_{k=2}^{\infty} \frac{(1-x)^k}{k}. \tag{5}$$

Since by assumption $1/\alpha - 1 < 1/2$, we have $\ln(1/\alpha) \geq (1/\alpha - 1)/2$. Substituting in (4), we get

$$E_1(\ln \frac{1}{\alpha}) \leq \ln \frac{1}{\frac{1-\alpha}{2\alpha}} + \frac{1}{2} = \ln \frac{2\alpha}{1-\alpha} + \frac{1}{2} \tag{6}$$

Substituting (6) into (2) gives us the following bound on sensitivity:

$$\|\mathbf{c}_0(x_1, \ldots, x_i, \ldots, x_T) - \mathbf{c}_0(x_1, \ldots, 1 - x_i, \ldots, x_T)\|_1 \leq \frac{1}{\alpha \ln 2} \left(\ln \frac{2\alpha}{1 - 1\alpha} + \frac{1}{2} + \ln 2 \right) \tag{7}$$

By Theorem 1 and (7), the algorithm ExponentialSum satisfies ε-differential privacy.

Accuracy. Clearly, $\mathrm{E}\hat{F}_e(j, \alpha) = F_e(j, \alpha)$. Next we upper bound σ^2, the maximum variance of $\hat{F}(j, \alpha)$ over all j. By Lemma 3, all intervals $[1, u_1], [u_1, u_2], \ldots, [u_r, j]$ correspond to nodes in distinct levels of $\mathcal{T}$, and therefore have sizes which are distinct powers of 2. We have, for some fixed constant C,

$$\sigma^2 \leq \left(C \frac{\log \frac{\alpha}{1-\alpha}}{\alpha \varepsilon} \right)^2 \sum_{i=1}^{\infty} \alpha^{2(2^i - 1)} = \left(C \frac{\log \frac{\alpha}{1-\alpha}}{\alpha \varepsilon} \right)^2 \frac{1}{\alpha^2} \sum_{i=2}^{\infty} \alpha^{2^i} \leq \frac{1}{\alpha^2} \left(C \frac{\log \frac{\alpha}{1-\alpha}}{\alpha \varepsilon} \right)^3.$$

The proof can be completed analogously to the proof of Theorem 3. □

4.4 Polynomial and Other Decay Functions

Unlike the running sum, window sum, or exponential decay sum problems, there is no easy way to combine polynomial decay sums over intervals $[a, b-1]$ and $[b, c]$ into a polynomial decay sum over $[a, c]$. Therefore, our techniques for estimating polynomial decay sum are considerably different and in fact apply to *arbitrary* slow-growing decay sum functions. We analyze the algorithm for polynomial decay and remark at the end of the section how it extends to arbitrary decay functions.

On a high level, we approximate the polynomial decay function $g(i) = (i+1)^{-c}$ by a function g' that is constant on exponentially growing in size intervals. Then we can approximate the decay sum induced by g' by running multiple instances of our window sum algorithm in parallel. This technique results in a bi-criteria approximation, i.e. our approximation guarantee has both a multiplicative and an additive approximation factor. As $c \to 1$ (i.e. as the range of the polynomial decay sum grows), the additive approximation factor remains bounded and is dominated by β^{-2}, where $(1 \pm \beta)$ is the multiplicative approximation factor. Thus the approximation guarantees for our algorithm are mostly determined by a trade-off between additive and multiplicative approximation.

For a given polynomial decay function $g = (i+1)^{-c}$ and the induced decay sum F, let us a fix a multiplicative error parameter β and define a function b as $\forall j \geq 1 : b(j) = \max\{i : g(i) \geq (1-\beta)^j\}$ and $b(0) = 0$. Intuitively $g(i)$ is almost constant for $i \in [b(j-1), b(j))$.

We can now define a function g' that approximates g: $\forall i \in [b(j-1), b(j)) : g'(i) = (1-\beta)^{j-1}$ Let F' be the decay sum induced by g'. From the definition of g' it is immediate that $\forall j, \forall \mathbf{x} \in \{0,1\}^j : (1-\beta)F(j) \leq F'(j) \leq F(j)$.

The PolynomialSum algorithm is shown as Algorithm 4. Note that we call the j-th instance of WindowSum with input consisting of time updates in $\{0, (1-\beta)^{j-1}\}$. It is straightforward to check that the WindowSum algorithm can handle such scaled instances without modification. Note also that we modify the WindowSum algorithm slightly by adjusting the magnitude of noise added to the variables associated with the dyadic trees kept by WindowSum.

Algorithm 4 PolynomialSum

Set $\lambda = \frac{\log(1/(1-\beta))}{c\beta^2} + \frac{1}{\beta}$.
Start an instance of WindowSum for input $x_1, \ldots$ with window size $W_1 = b(1)$ and initializing noise for each variable $\mathrm{Lap}(\lambda/\varepsilon)$. Set $j^* = 1$.
for all updates x_i **do**
  if $i = b(j^*) + 1$ **then**
    Set $j^* = j^* + 1$.
    Start a new instance of WindowSum with window size $W_{j^*} = b(j^*) - b(j^* - 1)$ and and initializing noise for each variable $\mathrm{Lap}(\lambda/\varepsilon)$.
  end if
  for all $k \leq j^*$ **do**
    Update the k-th instance of WindowSum with input $(1-\beta)^{k-1} x_{i-b(k-1)}$
  end for
  Output

$$\hat{F}_p(i, c) = \sum_{j \geq 0 : b(j) < i} F_w((1-\beta)^j x_1, \ldots, (1-\beta)^j x_{i-b(j)}, W_{j+1}).$$

end for

Theorem 6. PolynomialSum *satisfies ε-differential privacy, and for any j, with probability $1 - \gamma$, we have $(1 - \beta)F_p(j, c) - O(\delta) \leq \hat{F}_p(c) \leq F_p(j, c) + O(\delta)$, where*

$$\delta = \begin{cases} \frac{1}{\varepsilon} \left(\frac{1}{c\beta^2} \log \frac{1}{1-\beta} \right)^{1.5} \log^{0.5} \frac{1}{\gamma} & \text{if } \frac{1}{c\beta^2} \log \frac{1}{1-\beta} \geq \log \frac{1}{\gamma} \\ \frac{1}{\varepsilon} \frac{1}{c\beta^2} \log \frac{1}{1-\beta} \log \frac{1}{\gamma} & \text{if } \frac{1}{c\beta^2} \log \frac{1}{1-\beta} < \log \frac{1}{\gamma} \end{cases}$$

Furthermore, PolynomialSum *can be implemented to use $O(T)$ words of space and run in $O(\log^2 T / \log(1/(1-\beta)))$ time per update on inputs consisting of T updates.*

Proof. **Privacy.** The privacy analysis is analogous to

the analysis in the proof of Theorem 3, but we bound sensitivity over all instances of WindowSum. Due to the scaling of the input, the sensitivity of the j-th instance of WindowSum is bounded by $(1+\beta)^{j-1}(\log W_j+1)$. Let us first bound W_j. Observe that $b(j) = \lfloor g^{-1}((1-\beta)^j\rfloor$. For $g(i) = (i+1)^{-c}$, we have $b(j) \in [(1-\beta)^{-j/c} - 2, (1-\beta)^{-j/c} - 1]$. Then W_j can be bounded as $W_j = b(j) - b(j-1) \leq (1-\beta)^{-j/c} - (1-\beta)^{-(j-1)/c} + 1$. Since $1-\beta < 1$ and $j \geq 1$, we have $W_j \leq (1-\beta)^{-j/c}$. We can then bound the overall sensitivity is by

$$\sum_{j=1}^{\infty}(1-\beta)^{j-1}\log W_j + \sum_{j=0}^{\infty}(1-\beta)^j \leq \sum_{j=1}^{\infty}(1-\beta)^{j-1}\log\frac{1}{(1-\beta)^{j/c}} + \frac{1}{\beta} = \frac{1}{c\beta^2}\log\frac{1}{(1-\beta)} + \frac{1}{\beta} \quad (8)$$

Theorem 1 and (8) complete the privacy proof.

Accuracy. Note that $\mathrm{E}\hat{F}_p(j,c) = F'(j)$. The variance of $F_w((1-\beta)^j x_{b(j)}, \ldots, (1-\beta)^j x_k, W_j)$ is at most $2(1-\beta)^{2j}\lambda^2\log W_j$. Therefore, the total variance σ^2 of $\hat{F}_p(j,c)$ is

$$\sigma^2 \leq \lambda^2\frac{1}{c}\log\frac{1}{1-\beta}\sum_{j=0}^{\infty}(j+1)(1-\beta)^{2j} = \lambda^2\frac{1}{c\beta^2(2-\beta)^2}\log\frac{1}{1-\beta} = O\left(\left(\frac{1}{c\beta^2}\log\frac{1}{1-\beta}\right)^3\right)$$

Using Lemma 2 as in Theorem 3 we can show that for any j, with probability at least $1-\gamma$,

$$|\hat{F}_p(j,c) - F'(j)| = \begin{cases} O((\frac{1}{c\beta^2}\log\frac{1}{1-\beta})^{1.5}\log^{0.5}\frac{1}{\gamma}) & \text{if } \frac{1}{c\beta^2}\log\frac{1}{1-\beta} \geq \log\frac{1}{\gamma} \\ O(\frac{1}{c\beta^2}\log\frac{1}{1-\beta}\log\frac{1}{\gamma}) & \text{if } \frac{1}{c\beta^2}\log\frac{1}{1-\beta} < \log\frac{1}{\gamma} \end{cases}$$

Since for all $\mathbf{x}$ and all j, $(1-\beta)F(j) \leq F'(j) \leq F(j)$, this completes the proof. □

This algorithm can more generally be used to compute a private (under continual observation) approximation to a decayed sum F induced by a decay function g as long as g^{-1} grows subexponentially. In this case sensitivity remains bounded and the additive error guarantee is dominated by a function of β, but the exact function depends on g. The algorithm is not applicable to the window or running sum problem, since for them g^{-1} is not well defined; the guarantee for exponential decay sum is incomparable with the one in Theorem 5.

5. LOWER BOUNDS

We formalize the general packing-bound framework for lower bounding the dependence of the error δ on the error probability γ for algorithms that are private under continual observation and achieve (δ,γ)-utility. We also instantiate the framework with a construction that yields concrete lower bounds for the three decay sum problems considered in this paper. As far as the dependence on error probability is concerned, our lower bounds for window and exponential decay sums are tight. Our lower bound for polynomial decay sums is against a purely additive approximation and is not directly comparable to the upper bounds on the approximation factors of our algorithm.

Suppose that for a fixed error probability γ, we want to prove a lower bound on δ for any ε-differentially private algorithm that achieves (δ,γ)-utility with respect to a function $F(x_1,\ldots,x_j)$ We can take $\gamma = 2/3q$, and it follows, by the union bound, that for any set $Q \subseteq [n]$ of size q, with probability 2/3, the algorithm is within an absolute error δ from $F(x_1,\ldots,x_j)$ for all $j \in Q$. Assume that for some T we can construct $N+1$ instances $\mathbf{x}^0,\ldots,\mathbf{x}^N$, each of length T, that satisfy the following properties:

1. (Q,δ)-*independence*: for all $a,b \in \{0,\ldots,N\}, a \neq b$, there exists some $j \in Q \subseteq T$ such that $|F(x_1^a,\ldots,x_j^a) - F(x_1^b,\ldots,x_j^b)| > 2\delta$.
2. D-*closeness*: for all $a,b \in \{0,\ldots,N\}$, $\mathbf{x}^a$ and $\mathbf{x}^b$ have hamming distance at most D.

Lemma 4. *Assume there exists an ε-differentially private algorithm $\mathcal{A}$ that at time step j outputs $\hat{F}(x_1,\ldots,x_j)$. Assume further that for any $Q \subseteq \mathbb{N}$, $|Q| = q$, we have* $\Pr[\forall j \in Q : |\hat{F}(x_1,\ldots,x_j) - F(x_1,\ldots,x_j)| \leq \delta] \geq 2/3$. *If for some Q there exists a set $\{\mathbf{x}^0,\ldots,\mathbf{x}^N\}$ that simultaneously satisfies (Q,δ)-independence and D-closeness with respect to F, then* $D > \frac{\ln N + \ln 2}{\epsilon}$

Proof. Let $B(\mathbf{x}^i) = \{\mathbf{f} : |f_j - F(x_1^i,\ldots,x_j^i)| \leq \delta\}$. By assumption, $\Pr[(\hat{F}(x_1^i,\ldots,x_j^i))_{j=1}^T \in B(\mathbf{x}^i)] \geq 2/3$. Then, by the definition of differential privacy and D-closeness, we have

$$\forall i : \Pr[(\hat{F}(x_1^0,\ldots,x_j^0))_{j=1}^T \in B(\mathbf{x}^i)] \geq e^{-\varepsilon D}2/3.$$

By (Q,δ)-independence, $B(\mathbf{x}^a) \cap B(\mathbf{x}^b) = \emptyset$ for all $a \neq b$. Therefore,

$$\Pr[(\hat{F}(x_1^0,\ldots,x_j^0))_{j=1}^T \in \bigcup_{i=1}^N B(\mathbf{x}^i)] = \sum_{i=1}^N \Pr[(\hat{F}(x_1^0,\ldots,x_j^0))_{j=1}^T \in B(\mathbf{x}^i)] \geq Ne^{-\varepsilon D}2/3.$$

However, since $B(\mathbf{x}^0) \cap \bigcup_{i=1}^N B(\mathbf{x}^i) = \emptyset$, by the assumptions on $\mathcal{A}$ we have

$$\Pr[(\hat{F}(x_1^0,\ldots,x_j^0))_{j=1}^T \in \bigcup_{i=1}^N B(\mathbf{x}^i)] < 1/3.$$

Therefore, $2N < e^{\varepsilon D}$, and the lemma follows by taking logarithms. □

In order to apply Lemma 4, we need a method to construct a set of instances satisfying (Q,δ)-independence and D-closeness for a given error bound δ, such that D is upper bounded by a function of δ and N is lowerbounded by a function of $|Q|$. We show a construction that allows us to derive a lower bound for *any decayed sum problem*, where, naturally, the form of the lower bound depends on the specific problem, i.e. on the decay function g. As corollaries, we derive specific lower bounds for the problems we consider in this paper. In our construction, the set of vectors $\{\mathbf{x}^i\}_{i=0}^q$ is defined as $\mathbf{x}^0 = (0^{Dq})$ and $\mathbf{x}^i = (0^{(i-1)D}, 1^D, 0^{(q-i)D})$. We set $Q = \{j : D \text{ divides } j\}$ and choose δ according to the specific decay function g. Consider a general decayed sum function $F(x_1,\ldots,x_j)$ with a decay function g. The construction gives our main lower bound theorem.

THEOREM 7. *Assume there exists an ε-differentially private algorithm $\mathcal{A}$ that at time step j outputs $\hat{F}(x_1,\ldots,x_j)$ and achieves (δ,γ)-utility with respect to a decayed sum function F induced by g. Denote $G(x)=\sum_{i=0}^{x-1} g(i)$. Then $\delta \geq \frac{1}{2}G(\Omega(\frac{\log(1/\gamma)}{\varepsilon}))$.*

For the three problems considered in this paper we derive the following corollaries.

COROLLARY 1. *Assume there exists an ε-differentially private algorithm $\mathcal{A}$ that at time step j outputs $\hat{F}_w(j,W)$ and achieves (δ,γ)-utility with respect to $F_w(j,W)$. Then, $\delta \geq \Omega\left(\min\left\{\frac{W}{2},\frac{\log(1/\gamma)}{\varepsilon}\right\}\right)$.*

Note that the lower bound of [DPNR10] is a special case of the above corollary for $\gamma = 2/3W = 2/3T$.

COROLLARY 2. *Assume there exists an ε-differentially private algorithm $\mathcal{A}$ that at time step j outputs $\hat{F}_e(j,\alpha)$ and achieves (δ,γ)-utility with respect to $F_e(j,\alpha)$. Then, for $\alpha \in (2/3,1)$ we have $\delta \geq \Omega\left(\min\left\{\frac{\alpha}{1-\alpha},\frac{\log(1/\gamma)}{\varepsilon}\right\}\right)$.*

COROLLARY 3. *Assume there exists an ε-differentially private algorithm $\mathcal{A}$ that at time step j outputs $\hat{F}_p(j,c)$ and achieves (δ,γ)-utility with respect to $F_p(j,c)$. Then, $\delta \geq H_c(\Omega(\frac{\log(1/\gamma)}{\varepsilon})) \geq \Omega\left(1-\frac{\varepsilon^{c-1}}{\log^{c-1}(1/\gamma)}\right)$, where $H_c(k)$ is the k-th generalized harmonic number in power c.*

6. PRIVACY EXPIRATION

In some situations privacy concerns themselves are time-dependent. The knowledge of an event may be considered sensitive for a fixed period of time, after which the event is no longer relevant and privacy requirements may be relaxed. In this section we model this scenario by extending the definition of differential privacy under continual observation in a natural way. Our definition of privacy with expiration is satisfied by any algorithm which is private under continual observation; however, we investigate whether relaxing the privacy constraints for old events allows us to achieve stronger accuracy guarantees. We show that computing running sums under privacy with an expiration window W requires no more error than the window sum problem under standard privacy constraints. Thus privacy expiration leads to error that scales with the size of the expiration window rather than entire input. We observe a similar phenomenon for computing general linear maps online, for which we characterize the error complexity of output perturbation.

6.1 Model

DEFINITION 3. *Let $g:\mathbb{N}\to\mathbb{R}$ be a non-increasing function. Let $\mathcal{A}$ be a randomized online algorithm that at time step j outputs $\mathcal{A}(x_1,\ldots,x_j)$. $\mathcal{A}$ satisfies* ε-differential privacy with expiration g *if for all $T\in\mathbb{Z}$, for all measurable subsets S of the range of $\mathcal{A}$, all possible inputs $x_1,\ldots,x_T$, all $j\leq T$ and all $x'_j\in[0,1]$ (where probability is over the coin throws of $\mathcal{A}$)*

$$\Pr[(\mathcal{A}(x_1,\ldots,x_j,\ldots,x_k))_{k=1}^{T}\in S] \leq e^{g(T-j)\varepsilon}\Pr[(\mathcal{A}(x_1,\ldots,x'_j,\ldots,x_k))_{k=1}^{T}\in S].$$

When $g(i)$ takes the value 1 for $i<W$ and ∞ for $i\geq W$, we say $\mathcal{A}$ satisfies ε-differential privacy with expiration window W.

The expiration function g models how fast privacy concerns "degrade" with time. In the sequel, we focus on the privacy expiration window model that imposes a sharp threshold on how privacy degrades. However, smoother expiration models are possible.

To gain further intuition about privacy with expiration, let us consider a utility-theoretic implication of the definition. Assume each user has a cost function c_j associated with event x_j such that for each k, $c_j(\mathcal{A}(x_1,\ldots,x_k),k)$ gives the expected cost incurred at time k related to information about event x_j being revealed by the algorithm's output. We can model expiration of privacy concerns by a cost function of the form

$$c_j(\mathcal{A}(x_1,\ldots,x_k),k) = c'_j(\mathcal{A}(x_1,\ldots,x_k))g^{-1}(k-j).$$

Then an algorithm $\mathcal{A}$ that satisfies ε-differential privacy with expiration g has the property that reporting an event to the algorithm does not increase a user's cost by more than ε at any point in time, for any cost function c_j of the form above.

It is straightforward to check that privacy with expiration composes.

LEMMA 5. *If two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively satisfy ε_1 and ε_2-differential privacy with expiration g, then any function of their concatenated outputs will satisfy $(\varepsilon_1+\varepsilon_2)$-differential privacy with expiration g.*

6.2 Linear Maps

In this section, we focus on the class of online linear maps, which at every time step $i\in\mathbb{N}$, output the inner product $y_i=\mathbf{a}_i\mathbf{x}^i$, for some row vector $\mathbf{a}_i=(a_{i1},\ldots,a_{ii})$ of length i. Here $\mathbf{x}^i$ denotes the partial input $(x_1,\ldots,x_i)$. The entire linear map from time step 1 to time step T can be represented in vector form by $\mathbf{y}^T=\mathbf{A}^T\mathbf{x}^T$, where $\mathbf{A}^T$ is a square $N\times N$ lower-triangular matrix whose i-th row is given by $[\mathbf{a}_i,\mathbf{0}_{N-i}]$.

THEOREM 8. *Let $\mathcal{A}_{LM}$ be the (randomized) algorithm that at time step i outputs $\hat{y}_i=\mathbf{a}_i\mathbf{x}^i+z_i$, where $z_i\sim\mathrm{Lap}(0,\lambda_i)$. $\mathcal{A}_{LM}$ is ε-differentially private with expiration window W if the noise parameters $\{\lambda_i\}_{i\in[T]}$ satisfy the set of T linear constraints*

$$\forall j\in[T],\ \sum_{k=j}^{W+j-1}\frac{|a_{kj}|}{\lambda_k}\leq\varepsilon \tag{9}$$

PROOF. Consider the time step $t\in[T]$. For $j\in\{t-W+1,\ldots,t\}$, let $\mathbf{x}^t$ and $\mathbf{x}^t(j)$ denote neighboring input sequences which differ only by their j-th entries $x_j\neq x'_j$. Conditioned on $\mathbf{x}^t$, the output sequence at time t, $\hat{\mathbf{y}}^t=\mathbf{A}^t\mathbf{x}^t+\mathbf{z}^t$, has independent entries distributed according to the product distribution

$$f_{\mathbf{x}^t}=\mathrm{Lap}(\mathbf{a}_1\mathbf{x}^1,\lambda_1)*\ldots*\mathrm{Lap}(\mathbf{a}_t\mathbf{x}^t,\lambda_t).$$

Thus, $\forall j\in\{t-W+1,\ldots,t\}$, $\forall x_j\neq x'_j$, and $\forall s\in\mathbb{R}$,

$$\frac{f_{\mathbf{x}^t}(s)}{f_{\mathbf{x}^t(j)}(s)}=\prod_{k=1}^{N}\exp\left(\frac{|a_{kj}(x_j-x'_j)|}{\lambda_k}\right) =\exp\left(|x_j-x'_j|\sum_{k=j}^{N}\frac{|a_{kj}|}{\lambda_k}\right)\leq\exp(\varepsilon),\ \text{by (9)}. \tag{10}$$

This inequality holds for all $t\in[T]$. Thus Definition 3 is satisfied, which concludes the proof. □

Note that for each $i \in [T]$, the parameter λ_i appears only in W inequality constraints of the type (9), more precisely in the inequalities indexed by $j \in \{i-W+1,\dots,i\}$. A trivial solution would set all the noise parameters equal: $\lambda_k = \max_j \sum_{i=j}^{W+j-1} |a_{i,j}|$, for all $k \in [T]$. However this solution may be suboptimal in terms of accuracy. In the next theorem we exhibit a different feasible solution to the constraints (9) which achieves error that depends on the privacy window parameter W rather than the length of the input.

THEOREM 9. *The algorithm $\mathcal{A}_{LM}$ with noise parameters $\lambda_i = \frac{W}{\varepsilon} \max_{j\in\{i-W+1,\dots,i\}} |a_{ij}|$, achieves ε-differential privacy with expiration window W, and (δ,γ)-utility, with $\delta = O\left(\frac{W}{\varepsilon}\log(1/\gamma)C\right)$, for any $\gamma > 0$, where*

$$C = \max_{i\in[T]} \max_{j\in\{i-W+1,\dots,i\}} |a_{ij}|$$

.

PROOF. **Privacy**. For $j \in [T]$, we have

$$\sum_{k=j}^{W+j-1} \frac{|a_{kj}|}{\lambda_k} \le \sum_{k=j}^{W+j-1} \frac{\max_{i\in\{k-W+1,\dots,k\}} |a_{ki}|}{\lambda_k} = \sum_{k=j}^{W+j-1} \frac{\max_{i\in\{k-W+1,\dots,k\}} |a_{ki}|}{\max_{l\in\{k-W+1,\dots,k\}} |a_{kl}| W}\varepsilon = \varepsilon. \tag{11}$$

Thus, by Theorem 8, the algorithm is ε-differentially private with expiration window W.

Accuracy. For $i \in [T]$, let $\lambda = \frac{c}{\lambda_i}$, where c is a constant such that $\lambda < 0.75/\lambda_i$, and let $t = \frac{1.5\lambda^2\lambda_i^2+\log(2/\gamma)}{\sqrt{2}\lambda_i\lambda}$. Then

$$\begin{aligned} t\sigma_i = t\sqrt{2}\lambda_i &= \frac{1.5\lambda^2\lambda_i^2+\log(2/\gamma)}{\lambda} = 1.5c\lambda_i + \log(2/\gamma)\frac{\lambda_i}{c} \\ &= \log(2/\gamma)\frac{W}{\varepsilon} \max_{j\in\{i-W+1,\dots,i\}} |a_{ij}| \left(\frac{1.5c}{\log(2/\gamma)} + \frac{1}{c}\right) \\ &= O\left(\log(1/\gamma)\frac{W}{\varepsilon} \max_{j\in\{i-W+1,\dots,i\}} |a_{ij}|\right), \end{aligned} \tag{12}$$

$$\text{and, } \exp(0.75\lambda^2\sigma_i^2 - \lambda t\sigma_i) = \exp(1.5\lambda^2\lambda_i^2 - \lambda t\sqrt{2}\lambda_i) = \gamma/2. \tag{13}$$

Using (12) and (13) in Lemma 2, we get

$$\Pr\left[|\hat{y}_i - y_i| > O\left(\frac{W\log(1/\delta)}{\varepsilon} \max_{j\in\{i-W+1,\dots,i\}} |a_{ij}|\right)\right] \le \gamma. \tag{14}$$

This inequality holds for every $i \in [T]$. Note that $C \ge \max_{j\in\{i-W+1,\dots,i\}} |a_{ij}|$, for all $i \in [T]$. Thus,

$$\begin{aligned} &\Pr\left[|\hat{y}_i - y_i| > O\left(\frac{W\log(1/\delta)}{\varepsilon}C\right)\right] \\ &\le \Pr\left[|\hat{y}_i - y_i| > O\left(\frac{W\log(1/\delta)}{\varepsilon} \max_{j\in\{i-W+1,\dots,i\}} |a_{ij}|\right)\right], \end{aligned} \tag{15}$$

which together with (14) concludes the proof. □

COROLLARY 4. *For the running sum, ($a_{ij} = 1$, for all $i \ge j$, and 0 otherwise), algorithm $\mathcal{A}_{LM}$ with noise parameters $\lambda_i = \frac{W}{\varepsilon}$, for all $i \in [T]$, achieves ε-differential privacy with expiration window W, and (δ,γ)-utility, with $\delta = O\left(\frac{W}{\varepsilon}\log(1/\gamma)\right)$, for any $\gamma > 0$.*

In the following section we present an algorithm giving stronger accuracy guarantees for the running sum problem.

6.3 Running sum

Using Lemma 5 we derive a meta-algorithm. Given any ε-differentially private window sum algorithm $\mathcal{A}_W$ for window size W, Algorithm 5 computes running sum with privacy expiration window W and the same accuracy guarantees as $\mathcal{A}_W$. Using WINDOWSUM as a subroutine, we get an algorithm which is private with expiration window W and has additive error polylogarithmic in W.

Algorithm 5 PRIVACYEXPIRATION

Require: Algorithm $\mathcal{A}_W$ for window sum with window size W, ε-differentially private under continual observation. $\mathcal{A}_W$ computes window sum estimate $\hat{F}_w(i,W)$ at time step i.
for all inputs x_i **do**
  Compute $\sum_{k=1}^{i-W} x_k$.
  Simulate $\mathcal{A}_W$ with input x_i
  Output $\sum_{k=1}^{i-W} x_k + \hat{F}_w(i,W)$.
end for

THEOREM 10. *Let $\mathcal{A}_W$ satisfy ε-differential privacy and achieve (δ,γ)-utility. Then* PRIVACYEXPIRATION *satisfies ε-differential privacy with expiration window W and achieves (δ,γ)-utility.*

PROOF. **Privacy**. As usual, we argue privacy for any fixed arbitrary T. The output of PRIVACYEXPIRATION up to time T is a function of two sequences: $(\hat{F}_w(i,W))_{i=1}^T$ and $(x_{i-W})_{i=1}^T$. The first sequence satisfies ε-differential privacy by the assumption on $\mathcal{A}_W$, and, consequently, ε-differential privacy with expiration window W. On the other hand, the second sequence satisfies 0-differential privacy with expiration window W, since the sequence of outputs up to any time step T is not a function of any x_j for $j > T - W$. The privacy guarantee follows by composition.

Accuracy. Observe that each output is the sum of an exact (i.e. without added noise) sum and an approximate window sum $\hat{F}_w(i,W)$ for some i. Therefore PRIVACYEXPIRATION achieves accuracy guarantees no worse than those of $\mathcal{A}_W$ for window parameter W. □

7. EXTENSIONS AND APPLICATIONS

Algorithms for sum problems can be used to compute more sophisticated statistics as we described earlier. In this section we exhibit a few extensions and applications of our algorithms. We show how they can be used to compute sums over individual predicates and some special cases of sums over holistic predicates, including distinct counts which is of great interest. We also show how to compute histograms (over windows or decayed). In the following discussion we denote an arbitrary universe as $\mathcal{U}$.

7.1 Individual Predicates

We define an *individual predicate* abstractly as a function $\mathcal{P} : \mathcal{U} \to [0,1]$. Let the input at time step i be u_i, where $u_i \in \mathcal{U}$. The *decayed predicate sum* for an individual predicate $\mathcal{P}$ and decayed sum function F then is $F(\mathcal{P}(u_1),\dots,\mathcal{P}(u_j))$. Differential privacy and utility for predicate sums can be defined analogously to decayed sums. The following claim is immediate for individual predicates:

THEOREM 11. *Let $\mathcal{A}$ be an ε-differentially private algorithm that achieves (δ,γ)-utility with respect to a decayed sum F. Then, on input $\mathcal{P}(u_1),\ldots,\mathcal{P}(u_T)$, $\mathcal{A}$ is ε-differentially private with respect to $u_1,\ldots,u_T$ and achieves (δ,γ)-utility with respect to the decayed predicate sum for $\mathcal{P}$ and F.*

7.2 Holistic Predicate Sum

Individual predicates are limited in that they can depend only on a single update u_i rather than the whole sequence of updates. Here we define the more general notion of *holistic predicates* and treat the special case of low-sensitivity holistic predicates, with the distinct count problem as an important application.

A *holistic predicate* is a function $\mathcal{P}:\mathcal{U}^*\to[0,1]$. The decayed predicate sum for the holistic predicate $\mathcal{P}$ is $F(\mathcal{P}(u_1),\ldots,\mathcal{P}(u_1,\ldots,u_j))$.

Let us call a holistic predicate *k-sensitive* if for any sequence of updates $u_1,\ldots,u_T$, any $j\in[T]$ and any $u'_j\in\mathcal{U}$, the sequences $\mathcal{P}(u_1,\ldots,u_j),\ldots,\mathcal{P}(u_1,\ldots,u_j,\ldots,u_T)$ and $\mathcal{P}(u_1,\ldots,u'_j),\ldots,\mathcal{P}(u_1,\ldots,u'_j,\ldots,u_T)$ differ in at most k components. The following theorem follows from the basic properties of ε-differential privacy (proof omitted).

THEOREM 12. *Let $\mathcal{A}$ be an ε-differentially private algorithm that achieves (δ,γ)-utility with respect to a decayed sum F. Then, when given input $\mathcal{P}(u_1),\ldots,\mathcal{P}(u_1,\ldots,u_T)$ for a k-sensitive holistic predicate $\mathcal{P}$, $\mathcal{A}$ is $k\varepsilon$-differentially private with respect to $u_1,\ldots,u_T$ and achieves (δ,γ)-utility with respect to the decayed predicate sum for $\mathcal{P}$ and F.*

We can show that the fundamental *distinct count* problem can be encoded as a 2-sensitive holistic predicate. In the distinct element count problem the input is a sequence of updates $u_1,u_2,\ldots$, and at each time step j the goal is to approximate the number of distinct elements seen so far, i.e. $|\{u\in\mathcal{U}:\exists i\le j \text{ s.t. } u_i=u\}|$. This problem is equivalent to a predicate sum problem where F is simply the running sum function, and $\mathcal{P}(u_1,\ldots,u_j)$ is 0 when $\exists i<j: u_i=u_j$ and 1 otherwise. The proof of the following lemma is deferred to the full version of the paper.

LEMMA 6. *The predicate $\mathcal{P}(u_1,\ldots,u_j)=\mathbf{1}(\forall i<j: u_i\neq u_j)$ is 2-sensitive.*

Then, by Theorem 12 and the algorithm of Dwork et al. [DPNR10] for the running sum problem, we have the following result:

THEOREM 13. *There exists an ε-differentially private algorithm that achieves (δ,γ)-utility for the discrete element count problem with $\delta=O(\log^{1.5}T\log^{0.5}\frac{1}{\gamma})$ (in the case $\log T=\omega(\log\frac{1}{\gamma})$) or $\delta=O(\log T\log\frac{1}{\gamma})$ (in the case $\log T=O(\log\frac{1}{\gamma})$), where T is the number of updates.*

We leave open the problem of designing a private algorithm for estimating, at each time step, the number of distinct elements seen over the last W updates, with absolute error polylogarithmic in W.

7.3 Histograms

Consider a situation in which each update can belong to one of several categories. More formally, let the update at time step i be $(u_i,x_i)\in\mathcal{U}\times[0,1]$. Let $\mathbf{x}(u,j)$ be $\mathbf{x}$ restricted to all components x_i for $i\le j$ such that $u_i=u$. Then, at time step j, the algorithm outputs a vector $\mathbf{y}(j)\in\mathbb{R}^{\mathcal{U}}$, where $y_u(j)$ is an approximation to $F(\mathbf{x}(u,j))$, for some decayed sum function F. We call this the *decayed histogram* problem for F. Differential privacy under continual observation for decayed histogram problems can be defined analogously to decayed sum problems.

Given an algorithm to approximate a decayed sum, it can be easily extended to an algorithm for the corresponding decayed histogram problem.

THEOREM 14. *Let $\mathcal{A}$ be an ε-differentially private algorithm that achieves (δ,γ)-utility with respect to a decayed sum F. Then, there exists an ε-differentially private algorithm $\mathcal{A}'$ that uses $\mathcal{A}$ as a black box and for each j and each u satisfies $\Pr[|y_u(j)-F(\mathbf{x}(u,j))|>\delta]<\gamma$.*

8. CONCLUSION

We were inspired by the recent work on differential privacy of data analysis with continual updates [DPNR10, CSS10a], a research direction motivated by monitoring applications. However, our observation is that in monitoring applications typically recent data is more important than distant data. Hence, we need analyses that are accurate on the most recent window of data or data where past is decayed (polynomially or exponentially, as is common in database streaming).

We presented upper and lower bounds for a general class of functions — predicate sums — on window and decayed data. We derived our upper bounds by balancing noise at different levels of a tree atop the data in a nontrivial way, and derived lower bounds by packing arguments.

We introduced the "dual" problem when accuracy has to be with respect to the entire data of the past, but privacy guarantees expire, and presented differentially private algorithms whose accuracy scales with the privacy window. We reduced the running sum problem in this new model to computing window sum under the usual differential privacy constraints. We also characterized the error achieved by output perturbation for the general problem of computing a linear map online with privacy expiration.

Both the notions of decayed data and privacy expiration are important, and many aspects remain open for future study. Two concrete open problems are showing tight error lower bounds for privacy expiration, and reducing the running sum problem with smooth privacy expiration to decayed sums.

9. REFERENCES

[AS64] M. Abramowitz and I.A. Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, volume 55. Dover publications, 1964.

[CS03] E. Cohen and M. Strauss. Maintaining time-decaying stream aggregates. In *PODS*, 2003.

[CSS10a] T.H.H. Chan, E. Shi, and D. Song. Private and continual release of statistics. In *ICALP*, 2010.

[CSS10b] T.H.H. Chan, E. Shi, and D. Song. Private and continual release of statistics. Cryptology ePrint Archive, Report 2010/076, 2010.

[DGIM02] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows:(extended abstract). In *SODA*, 2002.

[DMNS06] C. Dwork, F. Mcsherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

[DPNR10] C. Dwork, T. Pitassi, M. Naor, and G Rothblum. Differential privacy under continual observation. In *STOC*, 2010.

[Dwo10] C. Dwork. Differential privacy in new settings. In *SODA*, 2010.

[War65] S. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of American Statistical Association*, 1965.