The Price of Differential Privacy under Continual Observation

Palak Jain[*]   Sofya Raskhodnikova[*]   Satchit Sivakumar[*]   Adam Smith[*]

January 12, 2022

Abstract

We study the accuracy of differentially private mechanisms in the continual release model. A continual release mechanism receives a sensitive dataset as a stream of T inputs and produces, after receiving each input, an accurate output on the obtained inputs. In contrast, a batch algorithm receives the data as one batch and produces a single output.

We provide the first strong lower bounds on the error of continual release mechanisms. In particular, for two fundamental problems that are widely studied and used in the batch model, we show that the worst case error of every continual release algorithm is $\tilde{\Omega}(T^{1/3})$ times larger than that of the best batch algorithm. Previous work shows only a polylogarithimic (in T) gap between the worst case error achievable in these two models; further, for many problems, including the summation of binary attributes, the polylogarithmic gap is tight (Dwork et al., 2010; Chan et al., 2010). Our results show that problems closely related to summation—specifically, those that require selecting the largest of a set of sums—are fundamentally harder in the continual release model than in the batch model.

Our lower bounds assume only that privacy holds for streams fixed in advance (the "nonadaptive" setting). However, we provide matching upper bounds that hold in a model where privacy is required even for adaptively selected streams. This model may be of independent interest.

Contents

1 Introduction

In fields ranging from healthcare to criminal justice, sensitive data is being analyzed to identify patterns and draw population-level conclusions. Differentially private (DP) data analysis [10] studies the design of algorithms that publish such aggregate statistics about input datasets while preserving the privacy of individuals whose data they contain. Differential privacy has been extensively studied and DP algorithms have been deployed in both industry and government. Current government deployments, notably at the US Census Bureau [6], operate in the *batch model*: that is, they collect their input all at once and produce a single output. However, in many situations, the data are collected over time, and the published statistics need to be updated regularly. An example of such a statistic is the number of COVID-19 cases. To investigate privacy in these situations, Dwork et al. [11] and Chan et al. [8] introduced the *continual release model*. In this model, a mechanism receives a sensitive dataset as a stream of T input records and produces, after receiving each record, an accurate output on the obtained inputs. Intuitively, the mechanism is differentially private if releasing the entire vector of T outputs satisfies differential privacy. The main challenge for privacy is that each individual record contributes to outputs at multiple time steps.

Dwork et al. [11] and Chan et al. [8] considered the problem of computing summation in the continual release model when each record consists of one bit. They designed a continual release mechanism, called the *binary tree mechanism*, that achieves (additive) error $O(\log^2 T)$ for this problem. Dwork et al. [11] also showed that an error of $\Omega(\log T)$ is necessary to privately release all running sums. (Further related work is discussed in Section 1.2.)

1.1 Our Contributions

We ask what price differentially private algorithms must pay in accuracy to solve a problem in the continual release model instead of the batch model. The largest previously known gap in accuracy between the two models is logarithmic in T, exhibited by the result of [11] on summation. We show that for two fundamental problems, which are related to summation and widely studied in the batch model, the gap is exponentially larger.

In the first problem, called MaxSum, each input consists of d binary attributes and the goal is to approximate the maximum of the attribute sums. We define the error of a mechanism as the maximum error over all the time steps. For MaxSum, the error at each time step is the absolute value of the difference between the true answer and the output of the mechanism at that time step. The second problem, SumSelect, is the "argmax" version of MaxSum: the goal is to find the index of the largest attribute sum. The error at a particular time step for this problem is the absolute difference between the maximum sum and the attribute sum at the index returned by the mechanism at that time step. Both problems are abstractions of practically relevant tasks. For instance, if the data collected by a public health agency (e.g., the US CDC) consists of records indicating which of d medical conditions each person suffers from, then MaxSum corresponds to the number of cases of the most common condition that occurred so far, and SumSelect corresponds to the name of this condition. Algorithms for these tasks are key ingredients in differentially private solutions to more complex problems such as synthetic data generation [16] and high-dimensional optimization [27]. We prove tight bounds on the error for these two problems in the continual release model in terms of the parameters T, called the *time horizon*, and d, called the *dimension*, discussed above, as well as the privacy parameter ε.

To provide a comparison to the continual release model, we assume here that algorithms in the batch model get input datasets of size T. Intuitively, a batch algorithm $\mathcal{A}$ is (ε, δ)-differentially private if, for all datasets $\mathbf{x}$ and $\mathbf{x}'$ that differ in one record, all events under the distributions $\mathcal{A}(\mathbf{x})$ and $\mathcal{A}(\mathbf{x}')$ have similar probabilities. In the case of $\delta = 0$ (also referred to as *pure differential privacy)*, these probabilities differ by at most a factor of e^ε. In the case of $\delta > 0$ (referred to as *approximate differential privacy*), if these probabilities are p and p', they must satisfy $p \leq e^\varepsilon \cdot p' + \delta$. (See Definition 2.2). To give a meaningful privacy guarantee, the parameter δ has to be small: in our case, $\delta = o(\varepsilon/T)$. For continual release mechanisms, we study *event-level* privacy, where each user's data appears in a single record, as opposed to *user-level* privacy, where a user's data could be distributed over multiple records. (See [11] for the discussion of these two variants.)

	Approximate DP ($\delta > 0$)	Pure DP ($\delta = 0$)	Reference
MaxSum	$\tilde{\Omega}\Big(\min\Big\{\sqrt[3]{\frac{T}{\varepsilon^2}}, \frac{\sqrt{d}}{\varepsilon}, T\Big\}\Big)$	$\tilde{\Omega}\Big(\min\Big\{\sqrt{\frac{T}{\varepsilon}}, \frac{d}{\varepsilon}, T\Big\}\Big)$	Thm. 3.1
	$\tilde{O}\Big(\min\Big\{\sqrt[3]{\frac{T}{\varepsilon^2}}, \frac{\sqrt{d}\ \text{polylog}(T)}{\varepsilon}, T\Big\}\Big)$	$\tilde{O}\Big(\min\Big\{\sqrt{\frac{T}{\varepsilon}}, \frac{d\ \text{polylog}(T)}{\varepsilon}, T\Big\}\Big)$	Cor. 5.6, 5.9
SumSelect	$\tilde{\Omega}\Big(\min\Big\{\sqrt[3]{\frac{T\log^2 d}{\varepsilon^2}}, \frac{\sqrt{d}}{\varepsilon}, T\Big\}\Big)$	$\tilde{\Omega}\Big(\min\Big\{\sqrt{\frac{T\log d}{\varepsilon}}, \frac{d}{\varepsilon}, T\Big\}\Big)$	Thm. 4.1
	$\tilde{O}\Big(\min\Big\{\sqrt[3]{\frac{T\log^2 d}{\varepsilon^2}}, \frac{\sqrt{d}\ \text{polylog}(T)}{\varepsilon}, T\Big\}\Big)$	$\tilde{O}\Big(\min\Big\{\sqrt{\frac{T\log d}{\varepsilon}}, \frac{d\ \text{polylog}(T)}{\varepsilon}, T\Big\}\Big)$	Cor. 5.6, 5.9

Table 1: Our results on the error of (ε, δ)-DP mechanisms in the continual release model. The corresponding upper and lower bounds differ only in the polylog(T) terms, highlighted in blue. For approximate differential privacy, the lower bounds apply when $\delta = o(\varepsilon/T)$, and the upper bounds apply when $\delta > \text{poly}\big(\frac{1}{T}\big)$.

We demonstrate a strong separation between the continual release and the batch models. For approximate differential privacy, we show that when d is sufficiently large, MaxSum_d and $\mathsf{SumSelect}_d$ require $\tilde{\Omega}(T^{1/3})$ and $\tilde{\Omega}\big(\big(\frac{T}{\log d}\big)^{1/3}\big)$ error blowup, respectively, in the continual release model compared to the batch setting. For pure differential privacy, the blowup (when d is large) is $\tilde{\Omega}(T^{1/2})$ for MaxSum and $\tilde{\Omega}(\frac{T^{1/2}}{\log^{1/2} d})$ for $\mathsf{SumSelect}$.

Our results are summarized in Table 1. To put our bounds in context, observe that for both problems we consider, there is a trivial algorithm that ignores its data, always outputs the same value and has error at most T (since each attribute sum is an integer between 0 and T). Our bounds on the error should be contrasted with the error achievable by $(\varepsilon, 0)$-differentially private algorithms in the batch model: $O(\frac{1}{\varepsilon})$ for MaxSum, and $O\big(\frac{\log d}{\varepsilon}\big)$ for $\mathsf{SumSelect}$. The former is obtained by an instantiation of the Laplace mechanism from [10] and the latter—by an instantiation of the exponential mechanism of McSherry and Talwar [22].

We obtain our lower bounds by reductions from problems in the batch model. The key is to consider tasks for which multiple instances of the same base problem on one dataset need to be solved. For MaxSum, the corresponding task in the batch model is to output all marginals. Each marginal can be thought of as an instance of computing the (appropriately rescaled) sum of values in the corresponding coordinate. For $\mathsf{SumSelect}$, the task in the batch model is based on solving independent instances of finding the largest marginal, each on its own subset of coordinates. We use the lower bounds for batch algorithms for these problems by Bun et al. [5], Hardt and Talwar [17], and Steinke and Ullman [26].

Each of our lower bounds is the minimum of three terms, corresponding to different parameter regimes. Our lower bounds are matched (up to polylogarithmic factors in T and $1/\delta$), in each regime, by two simple mechanisms and one trivial mechanism. The trivial mechanism always outputs an arbitrary value in the right range. The first simple mechanism is based on recomputing the value of the desired statistic (e.g., MaxSum) at regular intervals and providing the same answer until it is recomputed again. The second simple mechanism uses the binary tree mechanism to track all d coordinates separately and takes the maximum (or, in the case of $\mathsf{SumSelect}$, argmax) of the noisy values. The guarantees of these mechanisms for MaxSum, $\mathsf{SumSelect}$, and general functions of sensitivity 1 are stated in Section 5. Together, our mechanisms and our lower bounds characterize the error for MaxSum and $\mathsf{SumSelect}$ up to polylogarithmic factors in T and $1/\delta$ in all regimes.

Our lower bounds apply to the original continual release model of Dwork et al. [11] and Chan et al. [8]. In this model, which we refer to as the *nonadaptive* setting, privacy is defined for streams fixed in advance. However, our matching upper bounds hold even when privacy and accuracy are required for adaptively selected streams. In the adaptive version of the model, each record in the stream is chosen by an adversary after it sees all the answers of the mechanism from the prior time steps. This model gives more power to the adversary and therefore places more stringent requirements on privacy and accuracy. This model may be of independent interest.

1.2 Further Related Work

Event-level privacy Bolot et al. [3] and Perrier et al. [23] extended the tree mechanism of Dwork et al. [11] to work for weighted sums with exponentially decaying coefficients and for sums of bounded real values, respectively. Song et al. [25] generalized the model to graph data and obtained a continual release mechanism for graph statistics, such as the degree distribution and subgraph counts, on bounded degree graphs. Fichtenberger et al. [14] studied a variety of other graph problems in the continual release setting, including minimum cut and densest subgraph. Differentially private online learning is investigated in a sequence of works [19, 15, 1] that use the summation primitive developed by Dwork et al. to obtain sublinear regret guarantees for many hypothesis classes. The adaptive continual release model arises implicitly in those works, but to our knowledge it was not formulated explicitly. Cardoso and Rogers [7] study, among other problems, SumSelect (called *top-1 selection with unrestricted ℓ_0 sensitivity* in their work) in the continual release model. Their focus is on empirical performance on streams that arise in practice, in which the index of the largest sum changes seldom. The recomputation-based algorithm we present for SumSelect can be seen as a special case of their KnownBase algorithm. They evaluate the accuracy of the algorithm empirically whereas our work provides theoretical bounds on the error. One of the contributions of [7] is making the algorithms work in a more restrictive computational model, in which the algorithm only stores the current values of the sums at any given time step and the seed of a pseudorandom function. The algorithms we present here can also be implemented in their model using the techniques in their paper.

User-level differential privacy User-level privacy in the *continual release model* was first studied by Dwork et al. [11] and Chan et al. [8]. *User-level* privacy is more stringent than *event-level* privacy, so the lower bounds in our paper apply directly to that model. Even though, in general, event-level privacy does not imply user-level privacy, the recomputation technique used in some of our algorithms gives *user-level* privacy whenever the mechanism employed for the recomputations is user-level private.

Pan-Privacy Pan-privacy, defined by Dwork et al. [12], is a model that protects against intrusions into the memory of the algorithm as it processes a stream. In pan-privacy, as in continual release, the input is presented as a stream. However, the requirement of pan-privacy is orthogonal to that of continual release; see [12] for details.

2 Definitions

2.1 Preliminaries on Differential Privacy

We first introduce the notion of (ε, δ)-indistinguishability.

Definition 2.1 ((ε, δ)-Indistinguishability)**.** *Random variables R_1 and R_2 over the same outcome space $\mathcal{Y}$ are (ε, δ)-indistinguishable (denoted $R_1 \approx_{\varepsilon, \delta} R_2$) if for all subsets $S \subseteq \mathcal{Y}$, the following hold:*

$$\Pr[R_1 \in S] \leq e^{\varepsilon} \Pr[R_2 \in S] + \delta;$$
$$\Pr[R_2 \in S] \leq e^{\varepsilon} \Pr[R_1 \in S] + \delta.$$

A dataset $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{X}^n$ is a vector of elements, called *records*, from a universe $\mathcal{X}$. Two datasets are *neighbors* if they differ in one record (i.e., one coordinate). Informally, differential privacy requires that an algorithm's output distributions are similar on all pairs of neighboring datasets. In the batch model, the algorithm receives datasets as one batch as opposed to in an online fashion.

Definition 2.2 (Differential Privacy in Batch Model [10, 9])**.** *A randomized algorithm $\mathcal{A} : \mathcal{X}^n \to \mathcal{Y}$ is (ε, δ)-differentially private (DP) if for every pair of neighboring datasets $\mathbf{x}, \mathbf{x}' \in \mathcal{X}^n$,*

$$\mathcal{A}(\mathbf{x}) \approx_{\varepsilon, \delta} \mathcal{A}(\mathbf{x}').$$

The case $\delta = 0$ is referred to as pure *differential privacy*, whereas the case $\delta > 0$ is called approximate *differential privacy*.

Differential privacy protects groups of individuals.

Lemma 2.3 (Group Privacy [10]). *Every (ε, δ)-DP algorithm $\mathcal{A}$ is $(\ell\varepsilon, \delta')$-DP for groups of size ℓ, where $\delta' = \delta \frac{e^{\ell\varepsilon}-1}{e^{\varepsilon}-1}$; that is, for all datasets $\mathbf{x}, \mathbf{x}'$ such that $\|\mathbf{x} - \mathbf{x}'\|_0 \leq \ell$,*

$$\mathcal{A}(\mathbf{x}) \approx_{\ell\varepsilon, \delta'} \mathcal{A}(\mathbf{x}').$$

Differential privacy is closed under post-processing.

Lemma 2.4 (Post-Processing [10, 4]). *If $\mathcal{A}$ is an (ε, δ)-DP algorithm with output space $\mathcal{Y}$ and $\mathcal{B}$ is a randomized map from $\mathcal{Y}$ to $\mathcal{Z}$, then the algorithm $\mathcal{B} \circ \mathcal{A}$ is (ε, δ)-DP.*

Definition 2.5 (Sensitivity). *Let $f : \mathcal{X}^n \to \mathbb{R}^m$ be a function. Its ℓ_1-sensitivity is*

$$\max_{\text{neighbors } \mathbf{x}, \mathbf{x}' \in \mathcal{X}^n} \|f(\mathbf{x}) - f(\mathbf{x}')\|_1.$$

To define ℓ_2-sensitivity, we replace the ℓ_1 norm with the ℓ_2 norm.

Our algorithms use the standard Laplace mechanism to ensure differential privacy.

Definition 2.6 (Laplace Distribution). *The Laplace distribution with parameter b and mean 0, denoted $\mathsf{Lap}(b)$, has probability density*

$$h(r) = \frac{1}{2b} e^{-\frac{|r|}{b}} \text{ for all } r \in \mathbb{R}.$$

Lemma 2.7 (Laplace Mechanism). *Let $f : \mathcal{X}^n \to \mathbb{R}^m$ be a function with ℓ_1-sensitivity at most Δ_1. Then the Laplace mechanism is algorithm*

$$\mathcal{A}_f(\mathbf{x}) = f(\mathbf{x}) + (Z_1, \dots, Z_m),$$

where $Z_i \sim \mathsf{Lap}\left(\frac{\Delta_1}{\varepsilon}\right)$. Algorithm $\mathcal{A}_f$ is $(\varepsilon, 0)$-DP.

Lemma 2.8 (Exponential Mechanism [22]). *Let L be a set of outputs and $g : L \times \mathcal{X}^n \to \mathbb{R}$ be a function that measures the quality of each output on a dataset. Assume that for every $m \in L$, the function $g(m, .)$ has ℓ_1-sensitivity at most Δ. Then, for all $\varepsilon, n > 0$ and for all datasets $y \in \mathcal{X}^n$, there exists an $(\varepsilon, 0)$-DP mechanism that outputs an element $m \in L$ such that, for all $a > 0$, we have*

$$\Pr\left[\max_{i \in [L]} g(i, y) - g(m, y) \geq 2\Delta \frac{(\ln |L| + a)}{\varepsilon}\right] \leq e^{-a}.$$

Definition 2.9 (Gaussian Distribution). *The Gaussian distribution with parameter σ and mean 0, denoted $\mathcal{N}(0, \sigma^2)$, has probability density*

$$h(r) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{r^2}{2\sigma^2}} \text{ for all } r \in \mathbb{R}.$$

2.2 Preliminaries on ρ-zCDP

This section contains preliminaries about "zero-concentrated differential privacy" (zCDP). The difference between zero-concentrated differential privacy and (ε, δ)-differential privacy is that zCDP requires output distributions on all pairs of neighboring datasets to be ρ-close (Definition 2.11) instead of (ε, δ)-indistinguishable. In Section 5 we analyse the privacy of our upper bounds in terms of zCDP and then use the fact that zCDP implies (ε, δ)-differential privacy (Lemma 2.15) to compare our upper and lower bounds.

Definition 2.10 (Rényi Divergence [24]). *Let Q and Q' be distributions on $\mathcal{Y}$. For $\xi \in (1, \infty)$, the Rényi divergence of order ξ between Q and Q' (also called the ξ-Rényi Divergence) is defined as*

$$D_\xi(Q\|Q') = \frac{1}{\xi - 1} \log \left(\mathop{\mathbb{E}}_{r \sim Q'} \left[\left(\frac{Q(r)}{Q'(r)} \right)^{\xi-1} \right] \right). \tag{1}$$

Here $Q(\cdot)$ and $Q'(\cdot)$ denote either probability masses (in the discrete case) or probability densities (when they exist). More generally, one can replace $\frac{Q(.)}{Q'(.)}$ with the the Radon-Nikodym derivative of Q with respect to Q'.

Definition 2.11 (ρ-Closeness). *Random variables R_1 and R_2 over the same outcome space $\mathcal{Y}$ are ρ-close (denoted $R_1 \simeq_\rho R_2$) if for all $\xi \in (1, \infty)$,*

$$D_\xi(R_1\|R_2) \leq \xi\rho \text{ and } D_\xi(R_2\|R_1) \leq \xi\rho,$$

where $D_\xi(R_1\|R_2)$ is the ξ-Rényi divergence between the distributions of R_1 and R_2.

Definition 2.12 (zCDP in Batch Model [4]). *A randomized batch algorithm $\mathcal{A} : \mathcal{X}^n \to \mathcal{Y}$ is ρ-zero-concentrated differentially private (ρ-zCDP), if, for all neighboring datasets $\mathbf{y}, \mathbf{y}' \in \mathcal{X}^n$,*

$$\mathcal{A}(\mathbf{y}) \simeq_\rho \mathcal{A}(\mathbf{y}').$$

One major benefit of using zCDP is that this definition of privacy admits a clean composition result. We use it when analysing the privacy of the algorithms in Section 5.

Lemma 2.13 (Composition [4]). *Let $\mathcal{A} : \mathcal{X}^n \to \mathcal{Y}$ and $\mathcal{A}' : \mathcal{X}^n \times \mathcal{Y} \to \mathcal{Z}$ be batch algorithms. Suppose $\mathcal{A}$ is ρ-zCDP and $\mathcal{A}'$ is ρ'-zCDP. Define batch algorithm $\mathcal{A}'' : \mathcal{X}^n \to \mathcal{Y} \times \mathcal{Z}$ by $\mathcal{A}''(\mathbf{y}) = \mathcal{A}'(\mathbf{y}, \mathcal{A}(\mathbf{y}))$. Then $\mathcal{A}''$ is $(\rho + \rho')$-zCDP.*

The *Gaussian mechanism* is used in Section 5. It estimates a real-valued function on a database by adding Gaussian noise to the value of the function.

Lemma 2.14 (Gaussian Mechanism [4]). *Let $f : \mathcal{X}^n \to \mathbb{R}$ be a function with ℓ_2-sensitivity at most Δ_2. Let $\mathcal{A}$ be the batch algorithm that, on input $\mathbf{y}$, releases a sample from $\mathcal{N}(f(\mathbf{y}), \sigma^2)$. Then $\mathcal{A}$ is $(\Delta_2^2/2\sigma^2)$-zCDP.*

The final lemma in this section relates zero-concentrated differential privacy to (ε, δ)-differential privacy.

Lemma 2.15 (Conversion from zCDP to DP [4]). *For all $\rho, \delta > 0$, if batch algorithm $\mathcal{A}$ is ρ-zCDP, then $\mathcal{A}$ is $(\rho + 2\sqrt{\rho \log(1/\delta)}, \delta)$-DP.*

2.3 The Continual Release Model with Nonadaptively Chosen Inputs

A *mechanism* in the continual release model [11, 8] is an algorithm that receives its input $\mathbf{x} = (x_1, \ldots, x_T) \in \mathcal{X}^T$ as a stream. At each time step $t \in [T]$, it gets a record x_t and outputs an answer a_t. The output stream $(a_1, \ldots, a_T)$ is denoted by $\mathbf{a}$. We use $\mathbf{x}_{[t]} = (x_1, \ldots, x_t)$ for $t \in [T]$ to denote the first t records in a stream $\mathbf{x}$ (similarly, $\mathbf{a}_{[t]} = (a_1, \ldots, a_t)$.) The total number of records in the stream, denoted by T, is called the *time horizon*. For simplicity, we assume T is known to the mechanism.

We consider two variants of the continual release model. In the *nonadaptive* model of [11, 8], the input stream $\mathbf{x}$ is fixed before the mechanism runs. The *adaptive* model, defined in Section 5.1, allows an adversary to choose each input record x_t for $t \in \{2, \ldots, T\}$ based on the previous outputs $a_1, \ldots, a_{t-1}$ of the mechanism. The adaptive model gives the adversary more power. Therefore, the nonadaptive model provides weaker guarantees in terms of both privacy and accuracy. All our lower bounds are for the nonadaptive model and, consequently, imply the same lower bounds for the adaptive model. In contrast, all our algorithmic results are for the adaptive model (and, consequently, they also hold in the nonadaptive model).

We refer to standard algorithms that get their input in one batch and produce one output as *batch* algorithms. For clarity, we refer to continual release algorithms as mechanisms.

Accuracy   We start by defining how well a given output approximates the value of a function. We use a notion of error that depends on the function. Given a function $f : \mathcal{X}^* \to \mathcal{Y}$, a dataset $\mathbf{x} \in \mathcal{X}^*$, and an answer $a \in \mathcal{Y}$, let $\mathsf{ERR}_f(\mathbf{x}, a)$ be a nonnegative number that quantifies how far off a is from $f(\mathbf{x})$. Specifically, when $\mathcal{Y} = \mathbb{R}^k$,

$$\mathsf{ERR}_f(\mathbf{x}, a) = \|f(\mathbf{x}) - a\|_\infty. \tag{2}$$

Later (in (3)), we define a different notion of error for the optimization problem $\mathsf{SumSelect}$. Intuitively, the error for an optimization problem corresponds to the deficit in the objective function.

Definition 2.16 (Accuracy of a Mechanism). *In the nonadaptive continual release model, a mechanism $\mathcal{M}$ is (α, T)-accurate for f if, for all fixed input streams $\mathbf{x} = (x_1, \ldots, x_T)$, the maximum error $\mathsf{ERR}_f(\mathbf{x}_{[t]}, a_t)$ over the outputs $a_1, \ldots, a_T$ of mechanism $\mathcal{M}$ is bounded by α with high probability, that is,*

$$\Pr_{\substack{coins\ of\ \mathcal{M}}} \left[\max_{t \in [T]} \mathsf{ERR}_f(\mathbf{x}_{[t]}, a_t) \leq \alpha \right] \geq \frac{2}{3}.$$

Privacy   Finally, we define privacy in the nonadaptive continual release model.

Definition 2.17 (Privacy of a Mechanism). *Given a mechanism $\mathcal{M}$, define $\mathcal{A}_\mathcal{M}$ to be the batch model algorithm that receives an input dataset $\mathbf{x}$, runs $\mathcal{M}$ on stream $\mathbf{x}$, and returns the output stream $\mathbf{a}$ of $\mathcal{M}$. The mechanism $\mathcal{M}$ is (ε, δ)-differentially private (DP) in the nonadaptive continual release model if $\mathcal{A}_\mathcal{M}$ is (ε, δ)-DP in the batch model.*

Definition 2.17 refers to *event-level* privacy, where each user's data appears in a single record, as opposed to *user-level* privacy, where a user's data could be distributed over multiple records.

2.4   Problem Definitions

We consider two functions on datasets, where each record consists of d binary attributes. The first function, MaxSum_d, returns the maximum attribute sum for the input records. The second function, $\mathsf{SumSelect}_d$, returns the index of such a maximum sum.

Definition 2.18. *Let $d \in \mathbb{N}$ and $\mathcal{X} = \{0, 1\}^d$. For a dataset $\mathbf{x} \in \mathcal{X}^*$ and $j \in [d]$, the jth attribute of record x_i is its jth coordinate, denoted $x_i[j]$. Let $t \in \mathbb{N}$ and $\mathbf{x}_{[t]} \in \mathcal{X}^t$. The function $\mathsf{MaxSum}_d : \mathcal{X}^* \to \mathbb{N}$ is*

$$\mathsf{MaxSum}_d(\mathbf{x}_{[t]}) \stackrel{def}{=} \max_{j \in [d]} \left(\sum_{i \in [t]} x_i[j] \right).$$

The function $\mathsf{SumSelect}_d : \mathcal{X}^* \to [d]$ is

$$\mathsf{SumSelect}_d(\mathbf{x}_{[t]}) \stackrel{def}{=} \arg\max_{j \in [d]} \left(\sum_{i \in [t]} x_i[j] \right).$$

If multiple indices j attain the maximum sum, the function value is defined to be the smallest such index.

We study the accuracy of differentially private algorithms for computing these two functions. Our accuracy goal, stated in Definition 2.16, uses the notion ERR_f. We define the error $\mathsf{ERR}_{\mathsf{MaxSum}}$ as in (2). For $\mathsf{SumSelect}$, it is defined by:

$$\mathsf{ERR}_{\mathsf{SumSelect}}(\mathbf{x}_{[t]}, a_t) = \mathsf{MaxSum}_d(\mathbf{x}_{[t]}) - \sum_{i \in [t]} x_i[a_t]. \tag{3}$$

3   Lower Bounds for MaxSum

In this section, we prove Theorem 3.1 that provides strong lower bounds on the accuracy parameter α for any accurate mechanism for MaxSum_d in the nonadaptive continual release model. Our lower bounds match the upper bounds from Section 5 for MaxSum_d in the adaptive continual release model up to logarithmic factors in the time horizon T and the number of coordinates d.

Theorem 3.1. *For all $\varepsilon \in (0, 1], \delta \in [0, 1), \alpha \geq 0, d \in \mathbb{N}$, sufficiently large $T \in \mathbb{N}$, and mechanisms $\mathcal{M}$ in the nonadaptive continual release model that are (ε, δ)-differentially private and (α, T)-accurate for MaxSum_d, the following statements hold.*

1. If $\delta > 0$ and $\delta = o(\varepsilon/T)$, then $\alpha = \Omega\Big(\min\Big\{ \frac{T^{1/3}}{\varepsilon^{2/3}\log^{2/3}(\varepsilon T)}, \frac{\sqrt{d}}{\varepsilon\log d}, T \Big\} \Big)$.

2. If $\delta = 0$, then $\alpha = \Omega\Big(\min\Big\{ \sqrt{\frac{T}{\varepsilon}}, \frac{d}{\varepsilon}, T \Big\} \Big)$.

MaxSum_d can be released in the batch model with $\alpha = O(1/\varepsilon)$ via the Laplace mechanism [10]. Hence, Theorem 3.1 shows a strong separation between the batch model of differential privacy and the continual release model.

3.1   1-way Marginal Queries in Batch Model

To prove our lower bounds for MaxSum, we reduce from the problem of approximating 1-way marginals in the batch model. The function $\mathsf{Marginals}_d : \mathcal{X}^* \to [0, 1]^d$ maps a dataset $\mathbf{y}$ of any size n to a vector $(q_1(\mathbf{y}), \ldots, q_d(\mathbf{y}))$, where q_j, called the jth marginal, is defined as $q_j(\mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{y}[j]$. The error $\mathsf{ERR}_{\mathsf{Marginals}}$ is defined as in (2). Next, we define accuracy for batch algorithms.

Definition 3.2 (Accuracy of Batch Algorithms). *Let $\gamma \in [0, 1], n, d \in \mathbb{N}$, and $\mathcal{X} = \{0, 1\}^d$. Let $f : \mathcal{X}^n \to \mathbb{R}^d$ be a function on datasets. Batch algorithm $\mathcal{A}$ is (γ, n)-accurate for f if for all datasets $\mathbf{y} \in \mathcal{X}^n$,*

$$\Pr_{coins\ of\ \mathcal{A}}[\mathsf{ERR}_f(\mathbf{y}, \mathcal{A}(\mathbf{y})) \leq \gamma] \geq \frac{2}{3}.$$

We use the lower bounds from [5, 17] for the problem of estimating $\mathsf{Marginals}_d$ in the batch model. They are stated in Items 1 and 2 of Lemma 3.3 for approximate differential privacy and pure differential privacy, respectively. Item 2 in Lemma 3.3 is a slight modification of the lower bound from [17] and follows from a simple packing argument.

Lemma 3.3. *For all $\varepsilon \in (0, 1]$, $\delta \in [0, 1]$, $\gamma \in (0, 1)$, $d, n \in \mathbb{N}$, and algorithms $\mathcal{A}$ that are (ε, δ)-differentially private and (γ, n)-accurate for $\mathsf{Marginals}_d$, the following statements hold.*

    1 ([5]). *If $\delta > 0$ and $\delta = o(1/n)$, then $n = \Omega\left(\frac{\sqrt{d}}{\gamma\varepsilon\log d} \right)$.*

    2 ([17]). *If $\delta = 0$, then $n = \Omega\left(\frac{d}{\gamma\varepsilon} \right)$.*

3.2   Proof of Theorem 3.1

Let $\mathcal{M}$ be an (ε, δ)-DP and (α, T)-accurate mechanism for MaxSum_d in the nonadaptive continual release model. We use $\mathcal{M}$ to construct an (ε, δ)-DP batch algorithm $\mathcal{A}$ that is $(\frac{\alpha}{n}, n)$-accurate for $\mathsf{Marginals}_d$. The main idea in the construction, presented in Algorithm 1, is to force $\mathcal{M}$ to output an estimate of the sum for one attribute at a time by making the sum in that attribute the largest. First, $\mathcal{A}$ streams its own dataset $\mathbf{y}$ to $\mathcal{M}$. Then it sends n additional records with 1 in the first attribute and 0 everywhere else. After this, the first attribute sum is the largest, and the answer produced by $\mathcal{M}$ at this point can be used to estimate the first marginal. Then $\mathcal{A}$ equalizes the number of extraneous 1's for each attribute by sending n additional records with 0 in the first attribute and 1 everywhere else. It repeats this for each attribute, collecting the answers from $\mathcal{M}$, and then outputs its estimates for the marginals.

Algorithm 1 Algorithm $\mathcal{A}$ for estimating all 1-way marginals

---

Input: $\mathbf{y} = (y_1, \ldots, y_n) \in \mathcal{X}^n$, where $\mathcal{X} = \{0, 1\}^d$, and black-box access to mechanism $\mathcal{M}$.
    Output: $\mathbf{b} = (b_1, \ldots, b_d) \in \mathbb{R}^d$.
1: Let $\mathbf{e}_j$ be a vector of length d with 1 in coordinate j and 0 everywhere else; let $\overline{\mathbf{e}_j} \leftarrow (1)^d - \mathbf{e}_j$.
2: Construct a stream $\mathbf{x} \leftarrow \mathbf{y} \circ (\mathbf{e}_1)^n \circ (\overline{\mathbf{e}_1})^n \circ \cdots \circ (\mathbf{e}_{d-1})^n \circ (\overline{\mathbf{e}_{d-1}})^n \circ (\mathbf{e}_d)^n$ with $2dn$ records.
3: **for** $t \in [T]$ **do**
4:     Send x_t to $\mathcal{M}$ and get the corresponding output a_t.
5: **for** $j \in [d]$ **do**
6:     $b_j \leftarrow a_{2jn}/n - j$.
7: Output $\mathbf{b} \leftarrow (b_1, \ldots, b_d)$.

For vectors $\mathbf{u} = (u_1, \ldots, u_\ell)$ and $\mathbf{v} = (v_1, \ldots, v_m)$, let $\mathbf{u} \circ \mathbf{v} = (u_1, \ldots, u_\ell, v_1, \ldots, v_m)$. For a vector $\mathbf{v}$, let $\mathbf{v}^n$ denote the vector $\mathbf{v} \circ \mathbf{v} \circ \cdots \circ \mathbf{v}$ representing n concatenated copies of $\mathbf{v}$.

Lemma 3.4. *Let $\mathcal{A}$ be Algorithm 1. For all $\varepsilon > 0, \delta \geq 0, \alpha \in \mathbb{R}^+$ and $d, n, T \in \mathbb{N}$, where $T \geq 2dn$, if mechanism $\mathcal{M}$ is (ε, δ)-DP and (α, T)-accurate for MaxSum_d in the nonadaptive continual release model, then batch algorithm $\mathcal{A}$ is (ε, δ)-DP and $(\frac{\alpha}{n}, n)$-accurate for $\mathsf{Marginals}_d$.*

Proof. We start by reasoning about privacy. Fix neighboring datasets $\mathbf{y}$ and $\mathbf{y}'$ that are inputs to algorithm $\mathcal{A}$. Let $\mathbf{x}$ and $\mathbf{x}'$ be the streams constructed in Step 2 of $\mathcal{A}$ when it is run on $\mathbf{y}$ and $\mathbf{y}'$, respectively. By construction, $\mathbf{x}$ and $\mathbf{x}'$ are neighbors. Since $\mathcal{M}$ is (ε, δ)-DP, and $\mathcal{A}$ only post-processes the outputs received from $\mathcal{M}$, Lemma 2.4 implies that $\mathcal{A}$ is (ε, δ)-DP.

Now we reason about accuracy. Let $\mathbf{x} = (x_1, \ldots, x_{2dn})$ be the input stream provided to $\mathcal{M}$ when $\mathcal{A}$ is run on dataset $\mathbf{y}$. By construction of $\mathbf{x}$, the marginals $q_j(\mathbf{y})$ for all $j \in [d]$ and MaxSum_d are related as follows:

$$q_j(\mathbf{y}) = \frac{1}{n} \sum_{i \in [n]} y_i[j] = \frac{1}{n} \Big(\sum_{i \in [2jn]} x_i[j] - jn \Big) = \frac{1}{n} \cdot \mathsf{MaxSum}_d(\mathbf{x}_{[2jn]}) - j. \tag{4}$$

The attribute with the largest sum in $\mathbf{x}_{[2jn]}$ is j because $(\mathbf{e}_1)^n \circ (\overline{\mathbf{e}_1})^n \circ \cdots \circ (\mathbf{e}_{j-1})^n \circ (\overline{\mathbf{e}_{j-1}})^n \circ (\mathbf{e}_j)^n$ contributes jn ones to this attribute and $(j-1)n$ ones to each attribute in $[n]/\{j\}$, whereas the maximum sum of any attribute in $\mathbf{y}$ is n.

Since the transformation from $\mathcal{M}$ to $\mathcal{A}$ is deterministic, the coins of $\mathcal{A}$ are the same as the coins of $\mathcal{M}$. By (4) and the computation of the estimates for the $\mathsf{Marginals}_d$ in Step 6 of Algorithm 1,

$$\Pr_{\text{coins of } \mathcal{A}} \left[\mathsf{ERR}_{\mathsf{Marginals}}(\mathbf{y}, \mathcal{A}(\mathbf{y})) \leq \frac{\alpha}{n} \right] \qquad = \Pr_{\text{coins of } \mathcal{A}} \left[\max_{j \in [d]} |q_j(\mathbf{y}) - b_j| \leq \frac{\alpha}{n} \right]$$

$$= \Pr_{\text{coins of } \mathcal{M}} \left[\max_{t \in \{2n, \ldots, 2dn\}} |\mathsf{MaxSum}_d(\mathbf{x}_{[t]}) - a_t| \leq \alpha \right] \qquad \geq \Pr_{\text{coins of } \mathcal{M}} \left[\max_{t \in [T]} |\mathsf{MaxSum}_d(\mathbf{x}_{[t]}) - a_t| \leq \alpha \right]$$

$$= \Pr_{\text{coins of } \mathcal{M}} \left[\max_{t \in [T]} \mathsf{ERR}_{\mathsf{MaxSum}}(\mathbf{x}_{[t]}, a_t) \leq \alpha \right] \qquad \geq \frac{2}{3},$$

where we used that $\mathcal{M}$ is (α, T)-accurate for MaxSum_d. Thus, $\mathcal{A}$ is $(\frac{\alpha}{n}, n)$-accurate for $\mathsf{Marginals}_d$. $\square$

Now, we are ready to prove Theorem 3.1.

Proof of Theorem 3.1. Observe that the accuracy parameter α is nondecreasing as a function of d, since a mechanism $\mathcal{M}$ for MaxSum_d can be used to approximate $\mathsf{MaxSum}_{d'}$ for all $d' < d$ with the same accuracy and privacy guarantees by padding each length-d' input record with $d - d'$ zeroes.

Recall that both lower bounds on α stated in Theorem 3.1 are the minimum of three terms. To prove them, it suffices to show that, for all ranges of parameters, one of the terms is a lower bound on α.

First, consider the case when $\varepsilon \leq \frac{2}{T}$. We will show that in this case (for both pure and approximate differential privacy), $\alpha > T/9$. Since α is a nondecreasing function of d, it is sufficient to show this for $d = 1$.

10

Suppose for the sake of contradiction that $\alpha \leq T/9$. Let $\mathbf{x} = (0)^T$ and $\mathbf{x}' = (0)^{3T/4}(1)^{T/4}$ be datastreams that differ on $T/4$ records. Let a_T and a'_T be the final outputs of $\mathcal{M}$ on input streams $\mathbf{x}$ and $\mathbf{x}'$, respectively. By accuracy of $\mathcal{M}$, we have $\Pr[a_T \leq T/9] \geq 2/3$. Applying Lemma 2.3 on group privacy with $\varepsilon \leq 2/T$ and $\ell = T/4$, we get $\Pr[a'_T > T/9] \leq \sqrt{e} \cdot \Pr[a_t > T/9] + \frac{2\delta}{\varepsilon} < 2/3$ for sufficiently large T, since $\delta = o(\varepsilon/T)$. But $\mathsf{MaxSum}_d(\mathbf{x}') = T/4$, so $\mathcal{M}$ is not $(T/9, T)$-accurate, a contradiction. Hence, $\alpha = \Omega(T)$.

Now assume $\varepsilon > \frac{2}{T}$, i.e., $\varepsilon T > 2$. We start by proving Item 1 (when $\delta = o(\varepsilon/T)$). Let $\mathcal{A}$ be the algorithm for $\mathsf{Marginals}_d$ with black-box access to $\mathcal{M}$, as defined in Algorithm 1. If $T \geq 2dn$ and $\frac{\alpha}{n} < 1$, then by Lemma 3.4, algorithm $\mathcal{A}$ is (ε, δ)-differentially private and $(\frac{\alpha}{n}, n)$-accurate for $\mathsf{Marginals}_d$. (We require $\frac{\alpha}{n} < 1$ for the accuracy guarantee on $\mathcal{A}$ to be meaningful.) We can then use Lemma 3.3 to lower bound α.

Case 1: $d \leq (\varepsilon T \log(\varepsilon T))^{2/3}$. If there exists a dataset size $n \in (\alpha, \frac{T}{2d}]$, then by Item 1 of Lemma 3.3, $n = \Omega\left(\frac{n\sqrt{d}}{\alpha \cdot \varepsilon \log d}\right)$, and hence $\alpha = \Omega\left(\frac{\sqrt{d}}{\varepsilon \log d}\right)$. If no such n exists, then $\alpha + 1 \geq \frac{T}{2d}$, and hence $\alpha = \Omega(\frac{T}{d}) = \Omega\left(\frac{T^{1/3}}{\varepsilon^{2/3} \log^{2/3}(\varepsilon T)}\right)$. Combining the expressions for the two parameter ranges, we get that $\alpha = \Omega\left(\min\left\{\frac{T^{1/3}}{\varepsilon^{2/3} \log^{2/3}(\varepsilon T)}, \frac{\sqrt{d}}{\varepsilon \log d}\right\}\right)$.

Case 2: $d > (\varepsilon T \log(\varepsilon T))^{2/3}$. Set $d' = \lfloor(\varepsilon T \log(\varepsilon T))^{2/3}\rfloor$. Observe that $d \geq 1$ because $\varepsilon T > 2$. By our previous padding argument, a mechanism for MaxSum_d can be used to approximate $\mathsf{MaxSum}_{d'}$ for $d' = (\varepsilon T)^{2/3}$ with the same accuracy and privacy guarantees. Therefore, $\alpha = \Omega\left(\min\left\{\frac{T^{1/3}}{\varepsilon^{2/3} \log^{2/3}(\varepsilon T)}, \frac{\sqrt{d'}}{\varepsilon \log d'}\right\}\right) = \Omega\left(\frac{T^{1/3}}{\varepsilon^{2/3} \log^{2/3}(\varepsilon T)}\right)$. This completes the proof of Item 1.

The proof of Item 2 (with $\delta = 0$) proceeds along the same lines, except that we consider the cases $d \leq \sqrt{\varepsilon T}$ and $d > \sqrt{\varepsilon T}$ and use Item 2 from Lemma 3.3 instead of Item 1. If a dataset size $n \in (\alpha, \frac{T}{2d}]$ exists, by Item 2 of Lemma 3.3, we get $n = \Omega\left(\frac{nd}{\alpha\varepsilon}\right)$, and hence $\alpha = \Omega\left(\frac{d}{\varepsilon}\right)$. If no such n exists, then $\alpha + 1 \geq \frac{T}{2d}$, and hence $\alpha = \Omega(T/d) = \Omega(\sqrt{T/\varepsilon})$. If $d > \sqrt{\varepsilon T}$, a padding argument gives that $\alpha = \Omega(\sqrt{T/\varepsilon})$. $\qquad\square$

4 Lower Bounds for SumSelect

In this section, we prove Theorem 4.1 that provides strong lower bounds on the accuracy parameter α of any (α, T)-accurate algorithm $\mathcal{M}$ for $\mathsf{SumSelect}_d$ in the nonadaptive continual release model. Our lower bounds match the upper bounds from Section 5 for $\mathsf{SumSelect}_d$ in the adaptive continual release model up to logarithmic factors in the time horizon T and the number of coordinates d.

Theorem 4.1. *For all $\varepsilon \in (0, 1], \delta \in [0, 1), \alpha > 0, d \in \mathbb{N}$ such that $d > 1$, sufficiently large $T \in \mathbb{N}$, and mechanisms $\mathcal{M}$ in the nonadaptive continual release model that are (ε, δ)-DP and (α, T)-accurate for $\mathsf{SumSelect}_d$, the following statements hold.*

   1. If $0 < \delta = o(\frac{\varepsilon}{T})$, then $\alpha = \tilde{\Omega}\left(\min\left\{\frac{T^{1/3} \log^{2/3} d}{\varepsilon^{2/3}}, \frac{\sqrt{d}}{\varepsilon}, T\right\}\right)$.

   2. If $\delta = 0$, then $\alpha = \Omega\left(\min\left\{\sqrt{\frac{T}{\varepsilon} \log\left(2 + \frac{d}{\sqrt{\varepsilon T}}\right)}, \frac{d}{\varepsilon}, T\right\}\right) = \tilde{\Omega}\left(\min\left\{\sqrt{\frac{T \log(d)}{\varepsilon}}, \frac{d}{\varepsilon}, T\right\}\right)$.

4.1 k-Select$_d$ Problem in the Batch Model

To prove our lower bounds for $\mathsf{SumSelect}$ in the nonadaptive continual release model, we reduce from the problem called k-Select that solves k disjoint instances of the problem of selecting the index of the largest marginal in the batch model.

To define the function k-Select, let $n, d, k \in \mathbb{N}$, and $\mathcal{X} = \{0, 1\}^{kd}$. Let $\mathbf{y}[i : j]$ denote the dataset $\mathbf{y} \in \mathcal{X}^n$ with each record restricted to the coordinates between (and including) i and j. The function $k\text{-}\mathsf{Select}_d : \mathcal{X}^n \to [d]^k$ corresponds to dividing the dataset into k *blocks* $\mathbf{y}[1 : d], \mathbf{y}[d + 1 : 2d], \ldots, \mathbf{y}[(k - 1)d + 1 : kd]$, with n records each, and applying $\mathsf{SumSelect}_d$ independently on each block. It maps a dataset $\mathbf{y}$ of size n to a vector $(h_1(\mathbf{y}), \ldots, h_k(\mathbf{y}))$, where h_r is defined as the $\mathsf{SumSelect}_d$ function applied to block r:

$$h_r(\mathbf{y}) = \mathsf{SumSelect}_d\Big(\mathbf{y}\left[(r - 1)d + 1 : rd\right]\Big).$$

The accuracy for k-Select is defined as in Definition 3.2. To apply it, we define the error $\mathsf{ERR}_{k\text{-Select}}$. Note that the error is scaled differently than for SumSelect because the goal is to select the index of the largest marginal in each block, not of the largest sum. For $\mathbf{b} = (b_1, \ldots, b_k) \in [d]^k$, define $\mathsf{ERR}_{k\text{-Select}}(\mathbf{y}, \mathbf{b})$

$$= \max_{r \in [k]} \left(\frac{1}{n} \cdot \mathsf{ERR}_{\mathsf{SumSelect}}(\mathbf{y}[r(d-1)+1 : rd], b_r) \right).$$

Next, we state lower bounds for (ε, δ)-differentially private approximation of k-Select in the batch model.

Lemma 4.2. *For all $\varepsilon \in (0,1]$, $\delta \in [0,1]$, $\gamma \in [0, \frac{1}{20}]$, $d, k, n \in \mathbb{N}$, and batch algorithms $\mathcal{A}$ that are (ε, δ)-differentially private and (γ, n)-accurate for $k\text{-Select}_d$, the following statements hold.*

1. *If $\delta > 0$ and $\delta = o(1/n)$, then $n = \Omega(\frac{\sqrt{k} \cdot \log d}{\varepsilon \gamma \log(k+1)})$.*

2. *If $\delta = 0$, then $n = \Omega\left(\frac{k \cdot \log d}{\varepsilon \gamma} \right)$.*

Item 1 in Lemma 4.2 follows from Theorem 4.3 below.

Theorem 4.3 ([26, 28])**.** *For all $\varepsilon \in (0,1], \delta \in (0, 1/n), \gamma \in [0, \frac{1}{20}], d, n, k \in \mathbb{N}$, if Algorithm $\mathcal{A}$ is (ε, δ)-differentially private and (γ, n)-accurate for $k\text{-Select}_d$, then $n = \Omega(\frac{\sqrt{k} \log d}{\gamma \varepsilon \log(k+1)})$.*

Proof Sketch. We are aware of two proofs of this result, both of which were communicated to us by Jonathan Ullman [28]. The first uses the top-k selection lower bound of Steinke and Ullman [26]. In that problem, there is a single collection of d coordinates and the goal is to return the indices of $k < d$ coordinates whose sums are roughly largest.

For the specific distribution over instances that arises in the lower bound of [26], if one divides the coordinates into k equal groups, there is a constant probability that the collection of coordinates with the largest sum in each group is a good approximate solution for the top-k selection problem. An algorithm for $k\text{-Select}_d$ can thus be used to solve the top-k selection (out of dk coordinates) problem for such instances with roughly the same error and privacy parameter. The lower bound of [26] on n then applies.

Another approach is to use the composition framework of Bun, Ullman and Vadhan [5]. One can use a folklore result that selection among $d > 2^m$ coordinates can be used to mount a reconstruction attack on an appropriate dataset of size m. Composed with the lower bound for 1-way marginals in [5], one obtains a lower bound for $k\text{-Select}_d$. □

To complete the proof of Lemma 4.2, we prove Item 2 via a standard packing argument.

Proof of Item 2 in Lemma 4.2. For $\mathbf{u} \in [d]^k$, define $y_\mathbf{u}^* \in \{0,1\}^{dk}$ to be the record where each block $r \in [k]$ of d coordinates has a 1 in coordinate u_r and all zeros everywhere else. Let $\mathbf{y_u}$ be the dataset that consists of $2\gamma n$ copies of $y_\mathbf{u}^*$ and $(1-2\gamma)n$ copies of the all-zero record (assuming, for simplicity, that $2\gamma n$ is an integer). Since $\mathcal{A}$ is (γ, n)-accurate, $\Pr_{\text{coins of } \mathcal{A}}[\mathsf{ERR}_{k\text{-Select}}(\mathbf{y_u}, \mathcal{A}(\mathbf{y_u})) \leq \gamma] \geq \frac{2}{3}$ for all $\mathbf{u} \in [d]^k$. This means that for all $\mathbf{u} \in [d]^k$,

$$\Pr_{\text{coins of } \mathcal{A}}[\mathcal{A}(\mathbf{y_u}) = \mathbf{u}] \geq \frac{2}{3}.$$

For all $\mathbf{u}, \mathbf{u}' \in [d]$, by group privacy, $\mathcal{A}(\mathbf{y_u}) \approx_{(\gamma \varepsilon n, 0)} \mathcal{A}(\mathbf{y_{u'}})$, which implies that

$$\Pr[\mathcal{A}(\mathbf{y_u}) = \mathbf{u}'] \geq e^{-\gamma \varepsilon n} \Pr[\mathcal{A}(\mathbf{y_{u'}}) = \mathbf{u}'] \geq \frac{2}{3} e^{-\gamma \varepsilon n}. \tag{5}$$

Since the probability of any event is at most 1,

$$1 \geq \Pr_{\text{coins of } \mathcal{A}}[\mathcal{A}(\mathbf{y_u}) \neq \mathbf{u}] = \sum_{\mathbf{u}' \neq \mathbf{u}} \Pr[\mathcal{A}(\mathbf{y_u}) = \mathbf{u}'] \geq \frac{2}{3} e^{-\gamma \varepsilon n}(d^k - 1),$$

where the last inequality holds by (5). We get that $e^{\gamma \varepsilon n} \geq \frac{d^k - 1}{2} \cdot \frac{2}{3}$, and thus $n = \Omega\left(\frac{k \log d}{\gamma \varepsilon} \right)$. □

4.2 Proof of Theorem 4.1

Let $\mathcal{M}$ be an (ε, δ)-DP and (α, T)-accurate mechanism for SumSelect in the nonadaptive continual release model. We use $\mathcal{M}$ to construct an (ε, δ)-DP algorithm $\mathcal{A}$ that is $(\frac{\alpha}{n}, n)$-accurate for k-Select$_d$ in the batch model. We motivate our approach by first discussing an idea that doesn't quite work. Let $\mathcal{M}$ be an accurate mechanism for SumSelect$_d$ in the nonadaptive continual release model and $\mathbf{y}$ be a dataset with n records from $\{0, 1\}^{dk}$. A naive approach to solving k-Select$_d$ in the batch model is to run k instantiations of $\mathcal{M}$ for n time steps each, one on each block of d coordinates, to select the coordinate with the maximum sum in that block. However, running k instantiations of $\mathcal{M}$, as described, would result in a significant degradation of privacy, because every datapoint is used k times, once for each instantiation of $\mathcal{M}$. We instead reduce to SumSelect$_{dk}$ and run a single instantiation of $\mathcal{M}$ for about nk time steps, where each datapoint in $\mathbf{y}$ is sent to $\mathcal{M}$ only once. This approach doesn't suffer from privacy degradation.

Algorithm $\mathcal{A}$ proceeds in k stages; the rth stage is dedicated to selecting the coordinate with the maximum sum in the rth block. In the first stage, $\mathcal{A}$ streams $\mathbf{y}$ to $\mathcal{M}$. In order to select the coordinate with the maximum sum from the first block, $\mathcal{A}$ then sends $2n$ records of the form $(1^d 0^d \dots 0^d)$ to $\mathcal{M}$. Then the sums of the coordinates in the first block of $\mathbf{y}$ become much larger than the sums in the other blocks. This ensures that at the end of the first stage, $\mathcal{M}$ selects the coordinate with the maximum sum in the first block. In the second stage, $\mathcal{A}$ sends $2n$ records of the form $(0^d 1^d \dots 1^d)$ to $\mathcal{M}$ in order to balance out the number of extraneous 1's for each coordinate. In order to select the coordinate with the maximum sum from the second block, $\mathcal{A}$ sends $2n$ records of the form $(0^d 1^d 0^d \dots 0^d)$ to $\mathcal{M}$. At the end of the second stage, $\mathcal{M}$ selects the coordinate with the maximum sum in the second block. Algorithm $\mathcal{A}$ proceeds similarly for every block.

The details of the algorithm appear in Algorithm 2. For ease of indexing, $\mathcal{A}$ sends all-zero records in time steps $n + 1$ to $2n$ in Line 2 of Algorithm 2, to ensure that all stages have $4n$ time steps.

Algorithm 2 Batch algorithm $\mathcal{A}$ for k-Select

    Input: k, $\mathbf{y} = (y_1, \dots, y_n) \in \mathcal{X}^n$, where $\mathcal{X} = \{0, 1\}^{dk}$, and black-box access to mechanism $\mathcal{M}$.
    Output: $\mathbf{b} = (b_1, \dots, b_k) \in [d]^k$.
1: Let $\mathbf{v}_j$ be a vector of length dk with d ones in coordinates $[dj] \setminus [d(j-1)]$ and 0 everywhere else; let $\overline{\mathbf{v}_j} \leftarrow 1^{dk} - \mathbf{v}_j$.
2: Construct a stream $\mathbf{x} \leftarrow \mathbf{y} \circ (0^{dk})^n \circ (\mathbf{v}_1)^{2n} \circ (\overline{\mathbf{v}_1})^{2n} \circ \dots \circ (\mathbf{v}_{k-1})^{2n} \circ (\overline{\mathbf{v}_{k-1}})^{2n} \circ (\mathbf{v}_k)^{2n}$ with $4kn$ records.
3: **for** $t \in [T]$ **do**
4:     Send the record x_t to $\mathcal{M}$ and get the corresponding output a_t.
5: **for** $r \in [k]$ **do**
6:     $b_r \leftarrow a_{4rn} - d(r - 1)$. If $b_r \notin [d]$, then $b_r \leftarrow 1$.
7: Output $\mathbf{b} \leftarrow (b_1, \dots, b_k)$.

Lemma 4.4. *Let $\mathcal{A}$ be Algorithm 2. For all $\varepsilon > 0, \delta \geq 0$, $\alpha \in \mathbb{R}^+$, and $T, d, k, n \in \mathbb{N}$, where $T \geq 4kn$, if mechanism $\mathcal{M}$ is (ε, δ)-differentially private and (α, T)-accurate for SumSelect$_{dk}$ in the nonadaptive continual release model, then batch algorithm $\mathcal{A}$ is (ε, δ)-differentially private and $(\frac{\alpha}{n}, n)$-accurate for k-Select$_d$.*

Proof. We start by reasoning about privacy. Fix neighboring datasets $\mathbf{y}$ and $\mathbf{y}'$ that are inputs to algorithm $\mathcal{A}$. Let $\mathbf{x}$ and $\mathbf{x}'$ be the streams constructed in Step 2 of $\mathcal{A}$ when it is run on $\mathbf{y}$ and $\mathbf{y}'$, respectively. By construction, $\mathbf{x}$ and $\mathbf{x}'$ are neighboring streams. Since $\mathcal{M}$ is (ε, δ)-DP, and $\mathcal{A}$ only post-processes the outputs received from $\mathcal{M}$, Lemma 2.4 implies that $\mathcal{A}$ is (ε, δ)-DP.

Next, we reason about accuracy. Fix a dataset $\mathbf{y}$ and the corresponding data stream $\mathbf{x}$ sent to $\mathcal{M}$. Consider a setting τ of the random coins of $\mathcal{A}$. Since the transformation from $\mathcal{M}$ to $\mathcal{A}$ is deterministic, they correspond to coins used by $\mathcal{M}$ when $\mathcal{A}$ runs it as a subroutine. Let α_τ be the realized error of $\mathcal{M}$ with coins τ, that is,

$$\alpha_\tau = \max_{t \in [4kn]} \left(\mathsf{ERR}_{\mathsf{SumSelect}_{dk}}(\mathbf{x}_{[t]}, a_t) \right),$$

13

where a_t are the answers with coins τ. Similarly, let γ_τ be the realized error of $\mathcal{A}$ with coins τ, that is,

$$\gamma_\tau = \mathsf{ERR}_{k\text{-Select}_{dk}}(\mathbf{y}, \mathbf{b})$$
$$= \frac{1}{n} \cdot \max_{r \in [k]} \left(\mathsf{ERR}_{\mathsf{SumSelect}_d}(\mathbf{y}[(r-1)d + 1 : rd], b_r) \right),$$

where $\mathbf{b} = (b_1, \ldots, b_k)$ is the output of $\mathcal{A}$ run with coins τ.

The main observation in the accuracy analysis is that if α_τ is small, so is γ_τ. Note that if $\alpha \geq n$, the accuracy guarantee for $\mathcal{A}$ is vacuous. Now assume $\alpha < n$. For all blocks $r \in [k]$, the sums in $\mathbf{x}_{[4rn]} = \mathbf{y} \circ (0^{dk})^n \circ (\mathbf{v}_1)^{2n} \circ (\overline{\mathbf{v}_1})^{2n} \circ \cdots \circ (\mathbf{v}_{r-1})^{2n} \circ (\overline{\mathbf{v}_{r-1}})^{2n} \circ (\mathbf{v}_r)^{2n}$ of all coordinates not in block r are smaller than the sums of coordinates in block r by at least n. Consider coins τ with $\alpha_\tau \leq \alpha$. Since $\alpha_\tau < n$, the index a_{4rn} returned by $\mathcal{M}$ is in block r for all $r \in [k]$. Moreover, the error for each block is at most $\frac{\alpha_\tau}{n}$. Therefore, $\gamma_\tau \leq \frac{\alpha_\tau}{n} \leq \frac{\alpha}{n}$. Considering the probability of this event over all coins τ, we get

$$\Pr_{\text{coins } \tau \text{ of } \mathcal{A}} \left[\gamma_\tau \leq \frac{\alpha}{n} \right] \geq \Pr_{\text{coins } \tau \text{ of } \mathcal{M}} [\gamma_\tau \leq \alpha] \geq \frac{2}{3},$$

where the last inequality holds because $\mathcal{M}$ is (α, T)-accurate. We conclude that $\mathcal{A}$ is $(\frac{\alpha}{n}, n)$-accurate. $\qquad\square$

Finally, we prove Theorem 4.1.

Proof of Theorem 4.1. This proof's structure resembles that of Theorem 3.1. First, for the case of $\varepsilon \leq \frac{2}{T}$, we prove that $\alpha = \Omega(T)$. Let $\mathbf{e}_j$ be a record of length d with 1 in coordinate j and 0 everywhere else. Let $\mathbf{x} = (\mathbf{e}_1)^{T/4} \circ (0^d)^{3T/4}$ and $\mathbf{x}' = (\mathbf{e}_2)^{T/4} \circ (0^d)^{3T/4}$. Proceeding as in the proof of Theorem 3.1 (using group privacy and the error associated with selection) yields $\alpha = \Omega(T)$.

For all other values of ε, we reduce from k-Select, relying on the lower bounds for k-Select from Lemma 4.2. Fix T, d, ε. Given an integer k, the reduction of Lemma 4.4 maps a batch instance of k-Select$_{d'}$ of size n to an instance of SumSelect$_d$ with $d = d'k$ and $T = 4nk$. The reduction applies as long as $d' = \frac{d}{k} \geq 2$ and $n = \frac{T}{4k} \geq 1$ are integers. We will ignore the integrality requirement (which can be addressed by appropriate padding) and allow any k between 1 and $\min(\frac{d}{2}, \frac{T}{4})$.

When $\delta > 0$, the reduction leads to a lower bound on the error of $\min\left(\Omega\left(\frac{\sqrt{k}\log d'}{\varepsilon \log k}\right), n\right)$ when k and d' are sufficiently large constants. In our setting, this translates to a lower bound of $\Omega(\alpha_k)$ for $\alpha_k = \min\left(\frac{\sqrt{k}\log(2+d/k)}{\varepsilon \log(2+k)}, \frac{T}{k}\right)$. (We add 2 inside the logarithms to avoid 0 or subconstant log terms; this does not change the asymptotics.) Our goal is to select the value of $k \in [1, \min(\frac{d}{2}, \frac{T}{4})]$ that maximizes α_k. For fixed T, d, ε, let $k^* = k^*(T, d, \varepsilon) = \max(1, k')$ where k' denotes the largest value of k where the two terms defining α_k equalize (that is, k' satisfies $k'\sqrt{k'}\log(2+d/k')/\log(2+k') = \varepsilon T$). We use two basic facts about α_k: first, for $d > 15,000$, the function α_k is increasing on $[1, k^*)$ and decreasing on (k^*, ∞). Second, its maximum value α_{k^*} is $\tilde{\Omega}\left(\frac{T^{1/3}\log^{2/3}d}{\varepsilon^{2/3}}\right)$.

We consider four regimes for the triple (T, d, ε):

(a) $k^*(T, d, \varepsilon) = 1$: In this case, α_k is maximized at $k = 1$ and we obtain a lower bound of $\Omega(T/k) = \Omega(T)$.

(b) $k^*(T, d, \varepsilon) > \min(\frac{d}{2}, \frac{T}{4})$ and $2d \leq T$: In this case, we set $k = d/2$ and get a lower bound of $\alpha_k = \frac{\sqrt{k}\log(2+d/k)}{\varepsilon \log(2+k)}$ (since $k \leq k^*$), which is $\Omega\left(\frac{\sqrt{d}}{\varepsilon \log(2+d)}\right) = \tilde{\Omega}\left(\frac{\sqrt{d}}{\varepsilon}\right)$.

(c) $k^*(T, d, \varepsilon) > \min(\frac{d}{2}, \frac{T}{4})$ and $2d > T$: This case is not possible for large T. For it to occur, we must have $k^* > T/4$, which implies that $\alpha_{k^*} < 4$. Since $\alpha_{k^*} = \tilde{\Omega}\left(\frac{T^{1/3}\log^{2/3}d}{\varepsilon^{2/3}}\right)$, we get that $\varepsilon > 1$ (for sufficiently large T), contradicting our assumptions.

(d) $k^*(T, d, \varepsilon) \in [1, \min(\frac{d}{2}, \frac{T}{4})]$: In this case, we set $k = k^*$ and obtain a lower bound of $\alpha_{k^*} = \tilde{\Omega}\left(\frac{T^{1/3}\log^{2/3}d}{\varepsilon^{2/3}}\right)$.

Thus, for all possible relationships between T, d and ε, we obtain a lower bound that is one of three terms in the theorem statement.

The setting in which $\delta = 0$ is similar. For a given $k \in [1, \min(\frac{d}{2}, \frac{T}{4})]$, we obtain a lower bound of $\Omega(\alpha_k)$ for $\alpha_k = \min\left(\frac{k\log\left(2+\frac{d}{k}\right)}{\varepsilon}, \frac{T}{k}\right)$. The remaining calculations parallel the case where $\delta > 0$, except that now $\alpha_{k^*} = \Theta\left(\sqrt{\frac{T\log d}{\varepsilon}}\right)$. $\qquad\square$

5 Adaptive Upper Bounds

In this section, we define the adaptive continual release model and describe differentially private mechanisms for two types of problems in this model: $\mathsf{SumSelect}_d$ and approximating functions with bounded sensitivity (ℓ_2 sensitivity in the case of approximate differential privacy and ℓ_1 sensitivity in the case of pure DP). Our mechanisms are (α, T)-accurate, where the upper bounds for α match the lower bounds obtained in previous sections in the nonadaptive continual release model up to logarithmic factors in the time horizon T, the number of coordinates d, and the inverse of the privacy parameter $\frac{1}{\delta}$.

5.1 Adaptive Continual Release

In the adaptive continual release model, the input stream given to a mechanism $\mathcal{M}$ is chosen adversarially. That is, $\mathcal{M}$ interacts with a randomized adversarial process $\mathcal{A}dv$ that runs for T timesteps; at timestep $t \in [T]$, the process $\mathcal{A}dv$ receives a_t from $\mathcal{M}$, updates its internal state, and produces input record x_{t+1} that is sent to $\mathcal{M}$ at timestep $t + 1$. Process $\mathcal{A}dv$ can choose x_{t+1} based on the previous input records $\mathbf{x}_{[t]}$ and $\mathcal{M}$'s previous outputs $\mathbf{a}_{[t]}$. We make no assumptions on $\mathcal{A}dv$ regarding running time or complexity; its only limitation is that it does not see the internal coins of $\mathcal{M}$.

Definition 5.1. *A mechanism $\mathcal{M}$ is (α,T)-accurate for a function f in the adaptive continual release model if for all processes $\mathcal{A}dv$, the error of $\mathcal{M}$ with respect to $\mathcal{A}dv$ is at most α with high probability, that is,*

$$\Pr_{coins\ of\ \mathcal{M},\mathcal{A}dv}\left[\max_{t\in[T]}\mathsf{ERR}_f(a_t; \mathbf{x}_{[t]}) \le \alpha\right] \ge \frac{2}{3}.$$

A similar notion of accuracy was considered in work on adversarial streaming [2, 18, 20], though those articles do not directly address privacy.

Next, we define *(event-level) privacy in the adaptive continual release model* , which is trickier than in the nonadaptive continual release model . This concept is implicit in [15], but to our knowledge has not been previously defined. Privacy is defined with respect to the game $\Pi_{\mathcal{M},\mathcal{A}dv}$, described in Algorithm 3, between mechanism $\mathcal{M}$ and an adversary $\mathcal{A}dv$. In all timesteps except one, $\mathcal{A}dv$ outputs a single input record which $\Pi_{\mathcal{M},\mathcal{A}dv}$ simply forwards to $\mathcal{M}$. However, there is a special *challenge timestep* $t^* \in [T]$, selected by $\mathcal{A}dv$, in which $\mathcal{A}dv$ provides two records $x_{t^*}^{(L)}$ and $x_{t^*}^{(R)}$. The game comes in two versions, specified by its input parameter $\mathsf{side} \in \{L, R\}$ which is not known to $\mathcal{A}dv$ or $\mathcal{M}$: in one version, the record $x_{t^*}^{(L)}$ is handed to $\mathcal{M}$ at timestep t^*; in the other, the record $x_{t^*}^{(R)}$ is handed to $\mathcal{M}$ instead. The mechanism is private if the distributions on the adversary's view, which consists of its internal randomness and the transcript of messages it sends and receives, are close in the two versions of the game.

When the adversary decides in advance on all $T + 1$ records that it outputs over the course of the game, the resulting definition is equivalent to the nonadaptive version (Definition 2.17). The version we give here captures a richer class of settings.

Intuitively, we may think of $x_{t^*}^{(L)}$ as the data of person t^*, and of $x_{t^*}^{(R)}$ as a dummy value (say, all 0's). The parameter side then controls whether the data of person t^* is included in the computation or not. The privacy requirement is that an outside attacker cannot tell whether t^*'s data was used, even if the attacker has full knowledge of the process generating the data stream. The adversary $\mathcal{A}dv$ combines the data generation process and the attack itself in one entity, so that our model allows for an arbitrary relationship between them.

Algorithm 3 Privacy game $\Pi_{\mathcal{M},\mathcal{A}dv}$ for the adaptive continual release model

---

Input: time horizon $T \in \mathbb{N}$, side $\in \{L, R\}$ (not known to $\mathcal{A}dv$).

1: **for** $t = 1$ to T **do**
2:     $\mathcal{A}dv$ outputs $\mathsf{type}_t \in \{\mathsf{challenge}, \mathsf{regular}\}$, where $\mathsf{challenge}$ is chosen once during the game.
3:     **if** $\mathsf{type}_t = \mathsf{regular}$ **then**
4:         $\mathcal{A}dv$ outputs $x_t \in \mathcal{X}$ which is sent to $\mathcal{M}$.
5:     **if** $\mathsf{type}_t = \mathsf{challenge}$ **then**
6:         $t^* \leftarrow t$.
7:         $\mathcal{A}dv$ outputs $(x_t^{(L)}, x_t^{(R)}) \in \mathcal{X}^2$.
8:         $x_t^{(\mathsf{side})}$ is sent to $\mathcal{M}$.
9:     $\mathcal{M}$ outputs a_t which is given to $\mathcal{A}dv$.

Definition 5.2. *The* view of $\mathcal{A}dv$ in privacy game $\Pi_{\mathcal{M},\mathcal{A}dv}$ *consists of* $\mathcal{A}dv$'s *internal randomness and the transcript of messages it sends and receives. Let* $V_{\mathcal{M},\mathcal{A}dv}^{(\mathsf{side})}$ *denote* $\mathcal{A}dv$'s *view at the end of the game run with input* $\mathsf{side} \in \{L, R\}$.

One could also define the adversary's view as its internal state at the end of the game. The version we define contains enough information to compute that internal state, but is simpler to work with.

In addition to (ε, δ)-DP, we consider a related notion, called zCDP [4]. See Appendix 2.2 for background on zCDP and the notion of ρ-closeness of random variables ($\simeq_\rho$).

Definition 5.3. *A mechanism* $\mathcal{M}$ *is* (ε, δ)-DP *in the adaptive continual release model if, for all adversaries* $\mathcal{A}dv$,

$$V_{\mathcal{M},\mathcal{A}dv}^{(L)} \approx_{\varepsilon,\delta} V_{\mathcal{M},\mathcal{A}dv}^{(R)}.$$

A mechanism $\mathcal{M}$ *is* ρ-zCDP *in the adaptive continual release model if for all adversaries* $\mathcal{A}dv$,

$$V_{\mathcal{M},\mathcal{A}dv}^{(L)} \simeq_\rho V_{\mathcal{M},\mathcal{A}dv}^{(R)}.$$

The symbol $\simeq_\rho$ *denotes* ρ-closeness *(Definition 2.11).*

5.2 Statements of Adaptive Upper Bounds

In this subsection, we state theorems that summarize the performance guarantees of our mechanisms for MaxSum_d and $\mathsf{SumSelect}_d$. We prove these theorems in the following subsections. The upper bounds in these theorems are attained by two simple mechanisms: one uses the binary tree mechanism and the other recomputes the target function at regular intervals. We first state results for the binary-tree-based approach.

Theorem 5.4 (zCDP, Binary-Tree-Based Mechanisms). *For all* $\rho \in (0, 1]$, $d \in \mathbb{N}$, *and sufficiently large* $T > 0$, *there exist* ρ-zCDP *mechanisms* $\mathcal{M}, \mathcal{M}'$ *in the adaptive continual release model such that* $\mathcal{M}$ *is* (α, T)-accurate *for* MaxSum_d *and* $\mathcal{M}'$ *is* (α, T)-accurate *for* $\mathsf{SumSelect}_d$, *where* $\alpha = \mathrm{O}\left(\frac{\sqrt{d} \log T \sqrt{\log(dT)}}{\sqrt{\rho}}\right)$.

The next theorem uses the idea of recomputing at regular intervals, which applies quite generally. Item 1 of Theorem 5.5 applies for general sensitivity-1 functions (which include MaxSum_d); a similar result holds for bounded-sensitivity functions with output space $\mathbb{R}^d$.

Theorem 5.5 (zCDP, Mechanisms via Recomputing at Regular Intervals). *For all* $\rho \in (0, 1]$, $d \in \mathbb{N}$, *sufficiently large* $T > 0$, *and all functions* $f : \mathcal{X}^* \to \mathbb{R}$ *with* ℓ_2-sensitivity at most 1, *there exist* ρ-zCDP *mechanisms* $\mathcal{M}$ *and* $\mathcal{M}'$ *in the adaptive continual release model such that*

    1. Mechanism $\mathcal{M}$ *is* (α, T)-accurate *for* f *for* $\alpha = \mathrm{O}\left(\min\left\{\sqrt[3]{\frac{T \log T}{\rho}}, T\right\}\right)$;

16

2. *Mechanism $\mathcal{M}'$ is (α, T)-accurate for $\mathsf{SumSelect}_d$ for $\alpha = \mathrm{O}\left(\min\left\{\frac{T^{1/3}\log^{2/3}(dT)}{\rho^{1/3}}, T\right\}\right)$.*

Combining Theorems 5.4–5.5, using the conversion from zCDP to (ε, δ)-DP from Lemma 2.15 and substituting $\rho = \frac{\varepsilon^2}{16\log(1/\delta)}$, we get the following corollary.

Corollary 5.6. *For all $\varepsilon \in (0,1]$, $\delta \in (0, \frac{1}{2}]$, $d \in \mathbb{N}$, and sufficiently large $T > 0$, there exist (ε, δ)-DP mechanisms $\mathcal{M}$ and $\mathcal{M}'$ in the adaptive continual release model such that*

(1) $\mathcal{M}$ is (α, T)-accurate for MaxSum_d for $\alpha = \mathrm{O}\left(\min\left\{\frac{\sqrt[3]{T\log(1/\delta)\log T}}{\varepsilon^{2/3}}, \frac{\sqrt{d\log(dT)\log(1/\delta)}\,\log T}{\varepsilon}, T\right\}\right)$;

(2) $\mathcal{M}'$ is (α, T)-accurate for $\mathsf{SumSelect}_d$ for $\alpha = \mathrm{O}\left(\min\left\{\frac{\sqrt[3]{T\log^2(dT)\log(1/\delta)}}{\varepsilon^{2/3}}, \frac{\sqrt{d\log(dT)\log(1/\delta)}\,\log T}{\varepsilon}, T\right\}\right)$.

Simple variants of our mechanisms can be used to get the following theorems for $(\varepsilon, 0)$-differential privacy.

Theorem 5.7 (Pure DP, Binary-Tree-Based Mechanisms). *For all $\varepsilon \in (0,1]$, $d \in \mathbb{N}$, and sufficiently large $T > 0$, there exist $(\varepsilon, 0)$-DP mechanisms $\mathcal{M}$ and $\mathcal{M}'$ in the adaptive continual release model such that $\mathcal{M}$ is (α, T)-accurate for MaxSum_d and $\mathcal{M}'$ is (α, T)-accurate for $\mathsf{SumSelect}_d$ for $\alpha = \mathrm{O}\left(\frac{d(\log d)\log^3 T}{\varepsilon}\right)$.*

Theorem 5.8 (Pure DP, Mechanisms via Recomputing at Regular Intervals). *For all $\varepsilon \in (0,1]$, $d \in \mathbb{N}$, sufficiently large $T > 0$, and all functions $f : \mathcal{X}^* \to \mathbb{R}$ with ℓ_1-sensitivity at most 1, there exist $(\varepsilon, 0)$-DP mechanisms $\mathcal{M}$ and $\mathcal{M}'$ in the adaptive continual release model such that*

1. *Mechanism $\mathcal{M}$ is (α, T)-accurate for f for $\alpha = \mathrm{O}\left(\min\left\{\sqrt{\frac{T\log T}{\varepsilon}}, T\right\}\right)$;*

2. *Mechanism $\mathcal{M}'$ is (α, T)-accurate for $\mathsf{SumSelect}_d$ for $\alpha = \mathrm{O}\left(\min\left\{\sqrt{\frac{T\log(dT)}{\varepsilon}}, T\right\}\right)$.*

Theorems 5.7–5.8 yield the following corollary.

Corollary 5.9. *For all $\varepsilon \in (0,1]$, $d \in \mathbb{N}$, and sufficiently large $T > 0$, there exist $(\varepsilon, 0)$-DP mechanisms $\mathcal{M}$ and $\mathcal{M}'$ in the adaptive continual release model such that*

1. *$\mathcal{M}$ is (α, T)-accurate for MaxSum_d for $\alpha = \mathrm{O}\left(\min\left\{\sqrt{\frac{T\log T}{\varepsilon}}, T, \frac{d(\log d)\log^3 T}{\varepsilon}\right\}\right)$;*

2. *Mechanism $\mathcal{M}'$ is (α, T)-accurate for $\mathsf{SumSelect}_d$ for $\alpha = \mathrm{O}\left(\min\left\{\sqrt{\frac{T\log(dT)}{\varepsilon}}, T, \frac{d(\log d)\log^3 T}{\varepsilon}\right\}\right)$.*

5.3 Algorithms based on the Binary Tree Mechanism

In this section, we prove Theorem 5.4 for $\mathsf{SumSelect}_d$. Theorem 5.4 for MaxSum_d follows from the same analysis by considering the binary tree mechanism that outputs the highest noisy sum instead of the coordinate that achieves it.

In order to approximate $\mathsf{SumSelect}_d$ on a dataset with d attributes, we use the binary tree mechanism from [8, 11] to privately sum each of the attributes of the records $\mathbf{x}_{[t]}$ received so far, and then choose the attribute with the highest sum. For simplicity of exposition, in this section, we assume that T is a power of 2. In general, we can work with the smallest power of 2 greater than T. Throughout this section, $[i : j]$, where $i, j \in \mathbb{N}$, denotes the set of natural numbers $\{i, \ldots, j\}$.

At the high level, the binary tree mechanism constructs a complete binary tree with T leaves. The leaves correspond to the input records $\mathbf{x}_{[t]}$, where each record $x_i \in \{0,1\}^d$. Each internal node in the tree corresponds to the sum of all the leaves in its subtree. Each node stores the noisy version of the corresponding sum computed by adding a noise vector drawn from $\mathcal{N}(0, \sigma^2 \mathbb{I}^{d \times d})$ with $\sigma = \sqrt{\frac{d(\log T + 1)}{2\rho}}$. The algorithm that releases the noisy sum is $\frac{\rho}{\log T + 1}$-zCDP. Since each x_t participates in only $\log_2 T + 1$ sums in the tree, by

---

Algorithm 4 Mechanism $\mathcal{M}$ for $\mathsf{SumSelect}_d$ in adaptive continual release model

---

Input: time horizon $T \in \mathbb{N}$, privacy parameter ρ, stream $\mathbf{x} = (x_1, \ldots, x_T) \in \mathcal{X}^T$, where $\mathcal{X} = \{0,1\}^d$.

Output: stream $(a_1, \ldots, a_T) \in [d]^T$.

1: **Initialization:** Construct a complete binary tree with T leaves labeled $v_{[1:1]}, \ldots, v_{[T:T]}$. Label every internal node $v_{[\ell:r]}$ if the subtree rooted at that node has leaves $v_{[\ell:\ell]}, \ldots, v_{[r:r]}$. Initialize the partial sum $s_{[\ell:r]} \leftarrow 0^d$ for each node $v_{[\ell:r]}$ in the tree.

2: **for** $t = 1$ to T **do**

3:      Get record x_t from $\mathcal{A}dv$.

       ▷ *Compute noisy sums for nodes completed at time t*

4:      **for** each node $v_{[\ell:t]}$ **do**

5:          Draw noise $Z \sim \mathcal{N}(0, \sigma^2 \mathbb{I}^{d \times d})$, where $\sigma = \sqrt{\frac{d(\log T + 1)}{2\rho}}$, and set $s_{[\ell:t]} \leftarrow \sum_{i=\ell}^{r} x_i + Z$.

       ▷ *Output Steps:*

6:      $I_t \leftarrow$ collection of at most $\log t + 1$ disjoint intervals whose union is $[1 : t]$ and where each interval labels a node in the binary tree. (See Remark 5.3.)

7:      $\mathsf{sum}_t \leftarrow \sum_{[\ell:r] \in I_t} s_{[\ell:r]}$.

8:      Output $a_t \leftarrow \arg\max_{j \in [d]} \mathsf{sum}_t[j]$.

---

adaptive composition of zCDP (Lemma 2.13), the complete mechanism is ρ-zCDP (Theorem 5.4). The sum of all the attributes at any timestep can be calculated by adding at most $\log T$ of the sums stored in the tree, one at each level. The algorithm that adds the corresponding noisy sums is (α, T)-accurate for $\alpha \approx \mathrm{O}\left(\frac{\sqrt{d} \log T \log(Td)}{\sqrt{\rho}}\right)$. The formal description of the algorithm appears in Algorithm 4. The algorithm uses a dyadic decomposition (described in Remark 5.3) to decide which nodes of the tree it accesses to compute any particular output.

Remark (Dyadic Decomposition). *For any natural number $t > 1$, the interval $[1 : t]$ can be expressed as a union of at most $\log t + 1$ disjoint intervals as follows. Consider the binary expansion of t (which has at most $\log t + 1$ bits), and express t as a sum of distinct powers of 2 ordered from higher to lower powers. Then, the first interval $[1 : r]$ will have size equal to the largest power of 2 in the sum. The second interval will start at $r + 1$ and its size will be equal to the second largest power of 2 in the sum. Similarly, the remaining intervals are defined until all terms in the summation have been exhausted. For example, for $t = 7 = 4 + 2 + 1$, the intervals are $[1 : 4]$, $[5 : 6]$ and $\{7\}$.*

We present the privacy and accuracy analysis for Algorithm 4 in Lemmas 5.10 and 5.11, respectively, which together prove Theorem 5.4 for $\mathsf{SumSelect}$.

Lemma 5.10. *For all $\rho \in \mathbb{R}^+$, $d, T \in \mathbb{N}$, mechanism $\mathcal{M}$ described in Algorithm 4 is ρ-zCDP in the adaptive continual release model.*

Proof. Consider an adversary $\mathcal{A}dv$ interacting with the privacy game $\Pi_{\mathcal{M}, \mathcal{A}dv}$. We want to argue that the adversary's view is ρ-close in the two versions of the privacy game (for the two possible values of $\mathsf{side} \in \{L, R\}$.) We will achieve this by introducing a ρ-zCDP mechanism $\mathcal{M}_{\mathsf{gauss}}$ with input side and reducing our goal to the privacy of $\mathcal{M}_{\mathsf{gauss}}$.

For this, we use a simulation argument similar to those used in cryptography. Specifically, our proof defines two algorithms: (a) a ρ-zCDP mechanism $\mathcal{M}_{\mathsf{gauss}}$ that gets input $\mathsf{side} \in \{L, R\}$ and (b) a simulator $\mathcal{S}im$ with query access to $\mathcal{M}_{\mathsf{gauss}}$ that does not know the value of side. The simulator $\mathcal{S}im$ interacts with adversary $\mathcal{A}dv$ and satisfies a key guarantee:

> The view of the adversary $\mathcal{A}dv$ in its interaction with $\mathcal{S}im$ is identically distributed to its view in the privacy game $\Pi_{\mathcal{M}, \mathcal{A}dv}$, defined in Algorithm 3. (Figure 1 illustrates the structure of these two kinds of interaction.)

Algorithm 5 Simulator $\mathcal{S}$im for the proof of Lemma 5.10

    Input: time horizon $T \in \mathbb{N}$, privacy parameter $\rho \in \mathbb{R}^+$, black-box access to an adversary $\mathcal{A}dv$ and
         a mechanism $\mathcal{M}_{\mathsf{gauss}}$.
    Output: stream $(a_1, \ldots, a_T) \in [d]^T$.
    $\mathcal{A}dv$**:** At each timestep $t \in [T] \setminus \{t^*\}$, $\mathcal{A}dv$ provides $\mathcal{S}$im with record $x_t \in \mathcal{X}$, where $\mathcal{X} = \{0,1\}^d$. At
         the challenge timestep t^* (chosen by $\mathcal{A}dv$), it provides records $x_{t^*}^{(L)}, x_{t^*}^{(R)} \in \mathcal{X}$. At each timestep
         $t \in [T]$, $\mathcal{S}$im provides $\mathcal{A}dv$ with output $a_t \in [d]$.
    $\mathcal{M}_{\mathsf{gauss}}$**:** $\mathcal{S}$im exchanges $\log_2 T + 1$ messages with $\mathcal{M}_{\mathsf{gauss}}$.
1: **Initialization:** Perform Step 1 (the initialization phase) of Algorithm 4.
2: $j \leftarrow 1$.
3: **for** $t \in [T]$ **do**
4:     **if** $t = t^*$ **then**
5:         Get input $(x_{t^*}^{(L)}, x_{t^*}^{(R)})$ from $\mathcal{A}dv$.
6:         **for** $i \in [\log T + 1]$ **do**
7:             Send $(x_{t^*}^{(L)}, x_{t^*}^{(R)})$ to $\mathcal{M}_{\mathsf{gauss}}$, and get back a response p_i.
8:     **else**
9:         Get record x_t from $\mathcal{A}dv$.
10:     **for** each node $v_{[\ell,t]}$ **do**
11:         **if** $t^* \notin [\ell : t]$ where $[\ell : t]$ denotes the integers $\{\ell, ..., t\}$ **then**
12:             Draw noise $Z \sim \mathcal{N}(0, \sigma^2 \mathbb{I}^{d \times d})$, where $\sigma = \sqrt{\frac{d(\log T + 1)}{2\rho}}$.
13:             $s_{[\ell:t]} \leftarrow Z + \sum_{i=\ell}^{r} x_i$
14:         **else**
15:             $v_{[\ell:t]} \leftarrow \sum_{i \in [\ell:t] \setminus \{t^*\}} x_i + p_j$.
16:             $j \leftarrow j + 1$.
17:     *Output Steps:* Perform Steps 6–8 (the output steps) of Algorithm 4.

Algorithm 6 Mechanism $\mathcal{M}_{\mathsf{gauss}}$

    Input: $\mathsf{side} \in \{L, R\}$ (not known to $\mathcal{S}$im).
    Output: A natural number.
1: **for** $i = 1$ to $\log T + 1$ **do**
2:     Get records $v_i^{(L)}, v_i^{(R)} \in \{0,1\}^d$ from $\mathcal{S}$im.
3:     Draw noise from a multivariate Gaussian distribution $Z \sim \mathcal{N}(0, \sigma^2 \mathbb{I}^{d \times d})$, where $\sigma = \sqrt{\frac{d(\log T + 1)}{2\rho}}$.
4:     Output $v_i^{(\mathsf{side})} + Z$

Since the simulator's outputs to $\mathcal{A}dv$ are a post-processing of the query responses from $\mathcal{M}_{\mathsf{gauss}}$, we can argue that the adversary's view is ρ-close in the two versions of the privacy game $\Pi_{\mathcal{M}, \mathcal{A}dv}$.

To see why this is helpful, recall that we want to show that the probability of $\mathcal{A}dv$ guessing the value of side in the privacy game is small. If the probability of $\mathcal{A}dv$ guessing the value of side is the same in the privacy game as in its interaction with $\mathcal{S}$im, then—since the simulator doesn't know the value of side—$\mathcal{A}dv$ can only learn as much about side from its interaction with $\mathcal{S}$im as one can learn by querying $\mathcal{M}_{\mathsf{gauss}}$. Intuitively, if $\mathcal{M}_{\mathsf{gauss}}$ does not reveal much about the value of side then neither does $\mathcal{M}$. We now describe $\mathcal{M}_{\mathsf{gauss}}$ and the simulator, and formalize the argument.

The mechanism $\mathcal{M}_{\mathsf{gauss}}$ (described in Algorithm 6) gets an input $\mathsf{side} \in \{L, R\}$. It receives at most $\log T + 1$ queries of the form $v^{(L)}, v^{(R)}$ from $\mathcal{S}$im to which it responds with $p = v^{(\mathsf{side})} + Z$ where the noise Z is drawn from $\mathcal{N}(0, \sigma^2 \mathbb{I}^{d \times d})$ for $\sigma = \sqrt{\frac{d(\log T + 1)}{2\rho}}$. Observe that if $\mathcal{M}_{\mathsf{gauss}}$ has only a single interaction with $\mathcal{S}$im and outputs a single noised value, then by the privacy guarantee of the Gaussian mechanism (Lemma 2.14), $\mathcal{M}_{\mathsf{gauss}}$ is $\frac{\rho}{\log T + 1}$-zCDP. This can be seen by imagining that $\mathcal{M}_{\mathsf{gauss}}$ is computing a function
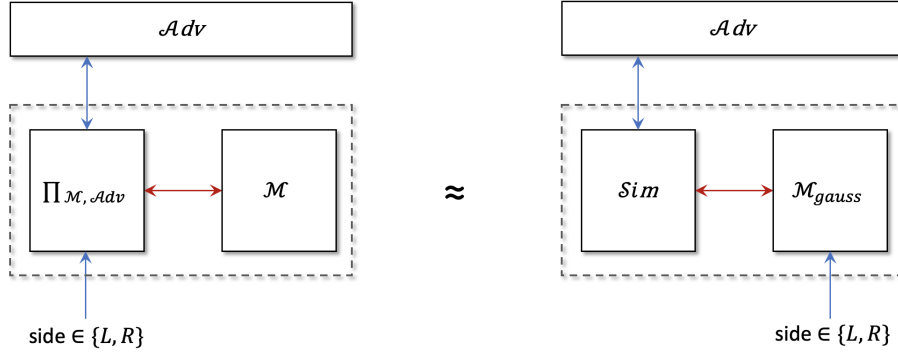
Figure 1: An illustration of the simulation argument from the proof of Lemma 5.10. The left-hand side shows the game used to define privacy with adaptively selected inputs. The right-hand side shows the simulation structure described in the proof. For each value of side, the adversary's view is identical in these two settings.

$f(\mathsf{side}) = x_i^{\mathsf{side}}$ and observing that the ℓ_2-sensitivity of f is $\sqrt{d}$. Since there are $\log T + 1$ interactions between $\mathcal{M}_{\mathsf{gauss}}$ and $\mathcal{S}$im, $\mathcal{M}_{\mathsf{gauss}}$ is an adaptive composition of $\log T + 1$ algorithms, each of which is $\frac{\rho}{\log T + 1}$-zCDP. By Lemma 2.13 on composition, $\mathcal{M}_{\mathsf{gauss}}$ is ρ-zCDP.

The simulator $\mathcal{S}$im (described in Algorithm 5) interacts with the adversary without knowing the input $\mathsf{side} \in \{L, R\}$ that is given to $\mathcal{M}_{\mathsf{gauss}}$. It queries $\mathcal{M}_{\mathsf{gauss}}$ exactly $\log T + 1$ times and uses the query responses to provide outputs to the adversary. The aim of the simulator is to mimic the behaviour of $\Pi_{\mathcal{M}, \mathcal{A}dv}$ even though it doesn't know side. The simulator constructs a binary tree as described in Algorithm 4. For all nodes in the binary tree except for those whose interval contains the challenge timestep t^*, the computation of the noisy subtree sums can be done by $\mathcal{S}$im without any help from $\mathcal{M}_{\mathsf{gauss}}$. For the nodes whose interval does contain t^*, the simulator sends $(x_{t^*}^{(L)}, x_{t^*}^{(R)})$ to $\mathcal{M}_{\mathsf{gauss}}$ and gets a noisy value of $x_{t^*}^{\mathsf{side}}$. It can then compute the corresponding subtree sum by adding the input records corresponding to the remaining leaves. Notice that the simulator can produce these outputs online—at the same time that $\Pi_{\mathcal{M}, \mathcal{A}dv}$ would.

The crucial point to note is that the view of the adversary $\mathcal{A}dv$ in the privacy game $\Pi_{\mathcal{M}, \mathcal{A}dv}$ is identically distributed to its view in the interaction with $\mathcal{M}_{\mathsf{gauss}}$ and $\mathcal{S}$im. Furthermore, the view of the adversary $\mathcal{A}dv$ when interacting with $\mathcal{S}$im and $\mathcal{M}_{\mathsf{gauss}}$ is simply a post-processing of the outputs provided to it by $\mathcal{S}$im, which are a post-processing of the outputs provided to $\mathcal{S}$im by $\mathcal{M}_{\mathsf{gauss}}$. Hence,

$$\mathcal{M}_{\mathsf{gauss}} \text{ is } \rho\text{-zCDP} \implies V_{\mathcal{M}, \mathcal{A}dv}^{(L)} \simeq_\rho V_{\mathcal{M}, \mathcal{A}dv}^{(R)}.$$

It remains to argue that exactly $\log T + 1$ nodes have a subtree sum that depends on the inputs from the challenge timestep t^*. Each node $v_{[\ell, r]}$ whose subtree sum depends on the inputs from timestep t^* satisfies $t^* \in [\ell : r]$. This holds only for one node at each level of the binary tree created by $\mathcal{S}$im (because the intervals represented by the nodes at a particular level are disjoint.) Since the binary tree has depth $\log T + 1$, exactly $\log T + 1$ nodes have a subtree sum that depends on the inputs from the challenge timestep t^*. $\quad\square$

Lemma 5.11. *For all $\rho > 0$ and sufficiently large $T \in \mathbb{N}$, mechanism $\mathcal{M}$ is (α, T)-accurate for* SumSelect *in the adaptive continual release model for* $\alpha = \mathrm{O}\left(\frac{\sqrt{d}\log T \sqrt{\log(Td)}}{\sqrt{\rho}}\right)$.

Proof. Consider any adversarial process $\mathcal{A}dv$ interacting with $\mathcal{M}$. We first argue that, at every timestep t, the random variable $\mathsf{ERR}_{\mathsf{SumSelect}_d}(\mathbf{x}_{[t]}, a_t)$ corresponding to the error at any timestep t can be upper bounded by a random variable that is the sum of at most $2\log t$ independent Gaussian random variables. We then use tail bounds for Gaussian random variables, along with a union bound, to argue that, with high probability,

the maximum value of this random variable is not too large. Finally, we take a union bound over timesteps to argue that, with high probability, $\max_{t \in [T]} \mathsf{ERR}_{\mathsf{SumSelect}_d}(\mathbf{x}_{[t]}, a_t)$ is not too large.

First, at any timestep t, let sum_t represent the vector of noisy sums defined in Step 7 of Algorithm 4. Therefore, each coordinate of this sum, $sum_t[j] = \sum_{i \in [t]} x_t[j] + \sum_{i \in [|I_t|]} Z_i$ is the sum of at most $\log t + 1$ noisy interval sums. Here, Z_i is a Gaussian random variable with mean 0 and standard deviation $\sigma = \sqrt{\frac{d(\log T + 1)}{2\rho}}$, and all Z_is are mutually independent. Hence, by the linearity of expectation, and by the linearity of the variance of independent random variables, we get that $\sum_{i \in [|I_t|]} Z_i$ is a Gaussian random variable with mean 0 and standard deviation $\sqrt{\frac{d(\log T + 1)|I_t|}{2\rho}} \leq \sqrt{\frac{d(\log T + 1)(\log t + 1)}{2\rho}}$. Consider the vector N consisting of the absolute values of d random variables independently drawn from the distribution $\mathcal{N}(0, \sigma^2)$ where $\sigma = \sqrt{\frac{d|I_t|(\log T + 1)}{2\rho}}$. The distribution of N is identical to the component-wise absolute values of the Gaussian noise vector $sum_t - \sum_{i \in [t]} x_i$. Then,

$$\sum_{i \in [t]} x_i[a_t] + \max_{j \in [d]} N[j] \geq \mathsf{MaxSum}_d(\mathbf{x}_{[t]}) - \max_{j \in [d]} N[j],$$

since if a_t is selected at timestep t, the noisy sum of coordinate a_t at timestep t is larger than the noisy sums of all other coordinates at timestep t (see Step 8 in Algorithm 4). Thus,

$$\mathsf{ERR}_{\mathsf{SumSelect}_d}(\mathbf{x}_{[t]}, a_t) = \mathsf{MaxSum}_d(\mathbf{x}_{[t]}) - \sum_{i \in [t]} x_i[a_t] \leq 2 \max_{j \in [d]} N[j].$$

Next, we reason about $\max_{j \in [d]} N[j]$ using standard probability tools. Set $\ell = \sqrt{10 \frac{d(\log T + 1)(\log t + 1) \log(dT)}{2\rho}}$. By Lemma A.2 on concentration of the maximum of the absolute values of Gaussian random variables, and since $\sigma \leq \sqrt{\frac{d(\log T + 1)(\log t + 1)}{2\rho}}$, we get that

$$\Pr\left[\max_{j \in [d]} N[j] > \ell\right] \leq 2de^{-\frac{\ell^2}{2\sigma^2}} \leq 2de^{-5 \log(dT)} \leq \frac{2}{T^5}.$$

Then, with probability at most $\frac{2}{T^5}$ (over the coins of the algorithm $\mathcal{A}$ and the adversarial process $\mathcal{A}dv$),

$$\mathsf{ERR}(\mathbf{x}_{[t]}, a_t) > 20\sqrt{\frac{d(\log T + 1)(\log t + 1) \log(dT)}{2\rho}}$$

$$\geq 20\sqrt{\frac{d(\log T + 1)^2 \log(dT)}{2\rho}},$$

since $t \leq T$. By a union bound over all $t \in [T]$, we get that $\max_{t \in [T]} \mathsf{ERR}_{\mathsf{SumSelect}_d}(\mathbf{x}_{[t]}, a_t) > 20\sqrt{\frac{d(\log T + 1)^2 \log(dT)}{2\rho}}$ with probability at most $\frac{2}{T^4} \leq \frac{1}{3}$ for sufficiently large T. This proves the lemma. $\square$

Proof Sketch of Theorem 5.7. The proof of Theorem 5.7 for $\mathsf{SumSelect}_d$ closely follows the exposition above. The mechanism used is the same as Algorithm 4, except that in Line 5, Z is drawn from $Lap(\frac{d(\log T + 1)}{\varepsilon})$ instead of a Gaussian distribution. The privacy proof is exactly as in Lemma 5.10, except that we use that the composition of $\log T + 1$ mechanisms that are $\left(\frac{\varepsilon}{\log T + 1}, 0\right)$-DP is $(\varepsilon, 0)$-DP instead a composition theorem for ρ-zCDP. The accuracy proof closely follows that of Lemma 5.11, with the main difference being that the the vector N is defined as the component-wise absolute value of d random variables independently drawn from the distribution of the sum of $|I_t|$ independent random variables distributed as $Lap\left(\frac{d(\log(T) + 1)}{\varepsilon}\right)$. We then use the concentration inequality for the maximum of the absolute values of independent Laplace random variables over $d|I_t|$ random variables in Lemma A.3 with $a = 2\log T$ to argue that the absolute value

of each Laplace random variable is smaller than $\frac{d(\log T+1)}{\varepsilon}(\log(d|I_t|)+10\log T)$ with probability at least $\frac{1}{T^{10}}$. This implies that $\max_{j\in[d]} N[j]$ is smaller than $\frac{d(\log T+1)^2}{\varepsilon}(\log(d(\log T+1))+10\log T)$ with probability at least $\frac{1}{T^{10}}$, upper bounding $|I_t|$ by $\log T+1$. Taking a union bound over T and using the fact that $\log(d(\log T+1)) \leq \log d(\log T+1)$ for sufficiently large T completes the proof. $\qquad\square$

Theorems 5.4 and 5.7 for MaxSum_d are proved analogously. The main difference is that we output $\max_{j\in[d]} sum_t[j]$ instead of $\arg\max_{j\in[d]} sum_t[j]$ in Line 8 of Algorithm 4.

5.4 Algorithms that Recompute at Regular Intervals

In this section, we prove Item 1 of Theorem 5.5 for sensitivity-1 functions. The proof of Item 2 of Theorem 5.5 builds on the same idea of recomputing $\mathsf{SumSelect}_d$ every T/m timesteps, but it uses the report noisy max (with exponential noise) algorithm for $\mathsf{SumSelect}_d$ [21] instead of adding Gaussian noise to the function. We omit the details, since the argument is essentially the same as in the rest of this section.

The mechanism recomputes the function every r timesteps. Between recomputations, it outputs the most recently computed value. We select r to balance the privacy cost of composition with the error due to returning stale values between recomputations.

Algorithm 7 Mechanism $\mathcal{M}$ for sensitivity-1 functions in adaptive continual release model

---

Input: time horizon T, privacy parameter $\rho > 0$, recompute period $r \in [T-1]$, function f, stream $\mathbf{x} = (x_1, \ldots, x_T) \in \mathcal{X}^n$ where $\mathcal{X} = \{0,1\}^d$.
Output: stream $(a_1, \ldots, a_T) \in \mathbb{R}^T$.
1: $m \leftarrow \lfloor \frac{T-1}{r} \rfloor$.
2: **for** $k = 1$ to m **do**
3:      Get input record $x_{(k-1)r+1}$.
4:      Draw $Z_k \sim \mathcal{N}(0, \sigma^2)$, where $\sigma = \sqrt{\frac{m}{2\rho}}$.
5:      Output $a_{(k-1)r+1} \leftarrow f(\mathbf{x}_{[(k-1)r+1]}) + Z_k$.
6:      **for** $t = (k-1)r + 2$ to kr **do**
7:          Get input record x_t.
8:          Output $a_t \leftarrow a_{(k-1)r+1}$.

Claim 5.12. *For all $\rho, T > 0$, $r \in [T-1]$, mechanism $\mathcal{M}$ defined in Algorithm 7 is ρ-zCDP in the adaptive continual release model.*

Proof. Consider an adversary $\mathcal{A}dv$ interacting with $\mathcal{M}$. We define a mechanism $\mathcal{M}_{\mathsf{comp}}$, similar to Algorithm 6, and a simulator $\mathcal{S}im$ that interacts with the adversary $\mathcal{A}dv$ such that the view of adversary $\mathcal{A}dv$ in the interaction with $\mathcal{M}_{\mathsf{comp}}$ and $\mathcal{S}im$ is identically distributed to its view in the privacy game $\Pi_{\mathcal{M},\mathcal{A}dv}$, defined in Algorithm 3.

Algorithm 8 Mechanism $\mathcal{M}_{\mathsf{comp}}$

Input: $\mathsf{side} \in \{L, R\}$ (not known to $\mathcal{S}im$)
Output: A natural number.
1: Get neighboring datasets $\mathbf{y}^{(L)}, \mathbf{y}^{(R)} \in \{0,1\}^d$ and a function f with ℓ_2 sensitivity at most 1 from $\mathcal{S}im$.
2: Draw noise $Z \sim \mathcal{N}(0, \sigma^2)$, where $\sigma = \sqrt{\frac{m}{2\rho}}$.
3: Output $f\left(y^{(\mathsf{side})}\right) + Z$

The mechanism $\mathcal{M}_{\mathsf{comp}}$ is defined in Algorithm 8. Since the function f has ℓ_2 sensitivity at most 1, then by the privacy of the Gaussian mechanism, and since the variance of the noise added is $\frac{m}{2\rho}$), $\mathcal{M}_{\mathsf{comp}}$ is $\frac{\rho}{m}$-zCDP with respect to the dataset consisting of $\mathsf{side} \in \{L, R\}$.

Algorithm 9 Simulator $\mathcal{S}$im for the proof of Claim 5.12

Input: time horizon T, privacy parameter $\rho > 0$, recompute period $r \in [T-1]$, function f. $\mathcal{S}$im also has black-box access to an adversary $\mathcal{A}dv$ and a process $\mathcal{M}_{\mathsf{comp}}$.

Output: stream $(a_1, \ldots, a_T) \in \mathbb{R}^T$

$\mathcal{A}dv$: At each timestep $t \in [T] \setminus \{t^*\}$, $\mathcal{A}dv$ provides $\mathcal{S}$im with record $x_t \in \mathcal{X}$. At the challenge timestep t^* (chosen by $\mathcal{A}dv$), it provides two records $x_{t^*}^{(L)}, x_{t^*}^{(R)} \in \mathcal{X}$. At every timestep $t \in [T]$, $\mathcal{S}$im provides $\mathcal{A}dv$ with output $a_t \in \mathbb{R}$.

$\mathcal{M}_{\mathsf{comp}}$: $\mathcal{S}$im exchanges T/r messages with $\mathcal{M}_{\mathsf{comp}}$.

1: **Initialization:** $m \leftarrow \lceil \frac{T}{r} \rceil$, $j \leftarrow 1$.
2: For timesteps $t < t^*$, run mechanism $\mathcal{M}$ in Algorithm 7 with inputs from $\mathcal{A}dv$, with the same T, ρ, r, f. Let a_t be $\mathcal{M}$'s output at timestep t.
3: **for** $t \geq t^*$ **do**
4:     **if** $t = t^*$ **then**
5:         Get input $(x_{t^*}^{(L)}, x_{t^*}^{(R)})$ from $\mathcal{A}dv$.
6:     **if** $t \neq t^*$ **then**
7:         Get record x_t from $\mathcal{A}dv$.
8:     **if** $t \bmod r = 1$ **then**
9:         Let $\mathbf{y}_t^{(\mathsf{side})} = \{x_1, \ldots, x_{t^*-1}, x_{t^*}^{(\mathsf{side})}, x_{t^*+1}, \ldots, x_t\}$ for each $\mathsf{side} \in \{L, R\}$
10:         $a_t \leftarrow \mathcal{M}_{\mathsf{comp}}\left(f, \mathbf{y}_t^{(L)}, \mathbf{y}_t^{(R)}\right)$.
11:     **else**
12:         $q \leftarrow \lfloor \frac{t}{r} \rfloor$; output $a_t \leftarrow a_{q+1}$.

The simulator $\mathcal{S}$im (described in Algorithm 9) gets inputs from $\mathcal{A}dv$, but it does not know the input $\mathsf{side} \in \{L, R\}$ that is given to $\mathcal{M}_{\mathsf{comp}}$. It interacts with $\mathcal{M}_{\mathsf{comp}}$ to provide outputs to the adversary $\mathcal{A}dv$. The aim of the Simulator is to mimic the behaviour of $\Pi_{\mathcal{M}, \mathcal{A}dv}$ even though it doesn't know side. For all timesteps $t < t^*$ before the challenge timestep, the simulator behaves exactly like $\mathcal{M}$. Starting at the challenge timestep, for every $t \in [t^* : T]$ where $\mathcal{M}$ would recompute the noised value of the sum, $\mathcal{S}$im sends $\mathcal{M}_{\mathsf{comp}}$ the function f as well as neighboring datasets $\mathbf{y}_t^{(L)}, \mathbf{y}_t^{(R)}$ defined by

$$\mathbf{y}_t^{(\mathsf{side})} = \{x_1, \ldots, x_{t^*-1}, x_{t^*}^{(\mathsf{side})}, x_{t^*+1}, \ldots, x_t\}.$$

Since $\mathcal{S}$im queries $\mathcal{M}_{\mathsf{comp}}$ at most m times, by adaptive composition, the output transcript of $\mathcal{M}_{\mathsf{comp}}$ is ρ-zCDP with respect to the dataset consisting of side.

The view of the adversary $\mathcal{A}dv$ in the real privacy game $\Pi_{\mathcal{M}, \mathcal{A}dv}$ is identically distributed to its view in the interaction with $\mathcal{M}_{\mathsf{comp}}$ and $\mathcal{S}$im. Furthermore, the view of the adversary $\mathcal{A}dv$ when interacting with $\mathcal{S}$im and $\mathcal{M}_{\mathsf{comp}}$ is simply a post-processing of the outputs provided to it by $\mathcal{S}$im, which are a post-processing of the outputs provided to $\mathcal{S}$im by $\mathcal{M}_{\mathsf{comp}}$. As argued previously, the output of $\mathcal{M}_{\mathsf{comp}}$ when $\mathsf{side} = L$ is ρ-close to its output transcript when $\mathsf{side} = R$. Hence we have that $V_{\mathcal{M}, \mathcal{A}dv}^{(L)} \simeq_\rho V_{\mathcal{M}, \mathcal{A}dv}^{(R)}$. $\qquad\square$

Claim 5.13. *Fix $\rho > 0$, sufficiently large $T > 0$, and $2 \leq m \leq T$. Let $f : \mathcal{X}^* \to \mathcal{Z}$ be a function with ℓ_2-sensitivity at most 1. Then mechanism $\mathcal{M}$, defined in Algorithm 7 is (α, T)-accurate for f in the adaptive continual release model where $\alpha = \frac{T}{m} + \sqrt{\frac{10m \log m}{\rho}}$.*

Proof. Consider any adversarial process $\mathcal{A}dv$ interacting with $\mathcal{M}$. Fix a timestep $t \in [T]$. Consider time horizon T divided into m stages, where the stage $k \in [m]$ is from timestep $(k-1)r + 1$ to kr. Let timestep t be in stage k. Intuitively, since $\mathcal{M}$, defined in Algorithm 7, corresponds to recomputing the noisy sum every r timesteps (and using each recomputed value for the next r timesteps), the error can be decomposed into two parts: one caused by the drift in the true value of the function since the last recomputation and the

other caused by noise addition. By the triangle inequality,

$$
\begin{aligned}
\mathsf{ERR}_f(\mathbf{x}_{[t]}, a_t) &= |a_t - f(\mathbf{x}_{[t]})| \\
&\leq |f(\mathbf{x}_{[t]}) - f(\mathbf{x}_{[(k-1)r+1]})| + |a_t - f(\mathbf{x}_{[(k-1)r+1]})| \\
&\leq T/m + |a_{(k-1)r+1} - f(\mathbf{x}_{[(k-1)r+1]})| \leq \frac{T}{m} + |Z_k|.
\end{aligned}
$$

The second inequality above holds because the ℓ_2-sensitivity of f is at most 1, and since we recompute every $r = T/m$ timesteps, the maximum change in the function f since the last recomputation is T/m. The third inequality follows from Steps 5 and 8 in Algorithm 7. Finally, observe that Z_k for $k \in [m]$ are mutually independent Gaussian random variables with mean 0 and standard deviation $\sqrt{\frac{m}{2\rho}}$. Hence, applying Lemma A.2 on the concentration of the maximum of the absolute values of Gaussian random variables (setting $\ell = \sqrt{\frac{10m \log m}{\rho}}$), and using the fact that $m \geq 2$,

$$
\Pr_{\text{coins of } \mathcal{A}, \mathcal{A}dv} \left(\max_{t \in [T]} \mathsf{ERR}_f(\mathbf{x}_{[t]}, a_t) \geq \frac{T}{m} + \sqrt{\frac{10m \log m}{\rho}} \right) = \Pr_{\text{coins of } \mathcal{A}, \mathcal{A}dv} \left(\max_{k \in [m]} |Z_k| \geq \sqrt{\frac{10m \log m}{\rho}} \right)
$$

$$
\leq \frac{2}{m^9} \leq \frac{1}{3}. \qquad \square
$$

Proof of Item 1 in Theorem 5.5. By Claim 5.12, the mechanism $\mathcal{M}$ is ρ-zCDP in the adaptive continual release model.

For $\rho \leq \frac{\log T}{T^2}$, consider the mechanism that doesn't touch the data and always outputs 0. Clearly it is 0-zCDP. Additionally, for this mechanism, $\alpha = O(T)$. For $\rho > \frac{\log T}{T^2}$, by Claim 5.13, mechanism $\mathcal{M}$ is (α, T)-accurate for f in the adaptive continual release model, where $\alpha = T/m + 10\sqrt{\frac{m \log m}{\rho}}$. Setting $m = \lfloor \frac{\rho^{1/3} T^{2/3}}{\log^{1/3} T} \rfloor$ gives $\alpha = O\left(\min \left\{ T, \sqrt[3]{\frac{T \log T}{\rho}} \right\} \right)$, where the min comes from the option of using the trivial mechanism. $\square$

Proof Sketch of Item 1 in Theorem 5.8. The mechanism $\mathcal{M}$ used is a variant of Algorithm 7. The only difference is that in Line 4, instead of the random variable Z_k being distributed as a Gaussian, it is distributed as $Lap(\frac{m}{\varepsilon})$. The privacy proof follows a structure similar to that of Claim 5.12, with the main difference being that instead of using a composition theorem for ρ-zCDP, we instead use that the composition of m mechanisms that are $(\frac{\varepsilon}{m}, 0)$-DP is $(\varepsilon, 0)$-DP.

For accuracy, we can prove a claim phrased exactly as Claim 5.13, with $\alpha = \frac{T}{m} + \frac{m}{\varepsilon}[\log m + 2 \log T]$ instead of $\alpha = \frac{T}{m} + \sqrt{\frac{10m \log m}{\rho}}$. The proof is similar, with the only difference being that instead of using Lemma A.2 on the maximum of i.i.d. Gaussian random variables, we instead use Lemma A.3 on the maximum of i.i.d. Laplace random variables, with $t = 2 \log T$.

Finally, we prove the theorem as follows: for $\varepsilon > \frac{\log T}{T}$, setting $m = \lfloor \sqrt{\frac{\varepsilon T}{\log T}} \rfloor$ in the accuracy claim gives $\alpha = O(\sqrt{\frac{T}{\varepsilon} \log T})$. For $\varepsilon \leq \frac{\log T}{T}$, we can consider the mechanism that always outputs 0 at every timestep. This mechanism is $(0, 0)$-DP and (α, T)-accurate for f in the adaptive continual release model with $\alpha = O(T)$. This completes the proof. $\square$

Proof Sketch of Item 2 in Theorems 5.5 and 5.8. We sketch the proof of Item 2 of Theorem 5.5. The proof of Item 2 of Theorem 5.8 is essentially the same. The upper bound mechanism $\mathcal{M}$ used for this proof is a variant of Algorithm 7 where we recompute $\mathsf{SumSelect}_d$ using the exponential mechanism [22] with $\varepsilon' = \sqrt{\frac{2\rho}{m}}$ (for Item 2 of Theorem 5.8 on pure DP, we use $\varepsilon' = \frac{\varepsilon}{m}$). The quality function of an attribute and dataset pair is defined to be the sum of that attribute over all entries in the dataset. The exponential mechanism instantiated as described above is used to privately compute $\mathsf{SumSelect}_d$ every T/m timesteps. Between recomputations, the attribute index produced at the last recomputation is used as the output.

The privacy proof follows a structure similar to that of Claim 5.12. The main difference for this proof is that the simulator will now interact with an ideal mechanism that takes as input a differentially private algorithm as well as neighboring datasets to run the algorithm on. In particular, the neighboring datasets will be the inputs $x_{t^*}^{(L)}$ and $x_{t^*}^{(R)}$ from the challenge timestep, and the algorithm will be the exponential mechanism hardcoded with all the inputs of the adversary so far (except for the inputs from the challenge timestep.) The ideal mechanism will run the algorithm with challenge input $x_{t^*}^{(\mathsf{side})}$ and output the result. The adversary's view in the privacy game is clearly identical to its view when interacting with the simulator. Finally, the closeness of the adversary's view in the simulated world when $\mathsf{side} = L$ and when $\mathsf{side} = R$ follows directly from the privacy of the exponential mechanism and adaptive composition [13, 4].

For accuracy, we prove a claim akin to Claim 5.13, with $\alpha = \frac{T}{m} + 2\sqrt{\frac{m}{2\rho}}[\log d + 5 \log m]$. The proof is similar to that of Claim 5.13; here, we define $|Z_k|$ as the error incurred by the kth instantiation of the exponential mechanism, and use Lemma 2.8 on the accuracy of the exponential mechanism (setting $a = 5 \log m$) and take a union bound over the m recomputations to argue that the maximum error is greater than $\alpha = \frac{T}{m} + 2\sqrt{\frac{m}{2\rho}}[\log d + 5 \log m]$ with probability at most $\frac{1}{m^4}$.

For $\rho > (\frac{\log(dT)}{T})^2$, by the accuracy claim, mechanism $\mathcal{M}$ is (α, T)-accurate for f in the adaptive continual release model, where $\alpha = \frac{T}{m} + 2\sqrt{\frac{m}{2\rho}}[\log d + 2 \log m]$. Setting $m = \lfloor \frac{\rho^{1/3} T^{2/3}}{(\log(dT))^{2/3}} \rfloor$ yields $\alpha = O\left(\frac{T^{1/3} \log(dT)^{2/3}}{\rho^{1/3}}\right)$. Finally, for $\rho \leq (\frac{\log(dT)}{T})^2$, consider the mechanism that doesn't touch the data and always outputs 0. It is clearly 0-zCDP, and has $\alpha = O(T)$. $\qquad\square$

Acknowledgments

References

[1] Naman Agarwal and Karan Singh. The price of differential privacy for online learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 32–40. PMLR, 06–11 Aug 2017.

[2] Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS'20, page 63–80, New York, NY, USA, 2020. Association for Computing Machinery.

[3] Jean Bolot, Nadia Fawaz, S. Muthukrishnan, Aleksandar Nikolov, and Nina Taft. Private decayed predicate sums on streams. In *Proceedings of the 16th International Conference on Database Theory*, ICDT '13, page 284–295, New York, NY, USA, 2013. Association for Computing Machinery.

[4] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 635–658, 2016.

[5] Mark Bun, Jonathan Ullman, and Salil Vadhan. Fingerprinting codes and the price of approximate differential privacy. *SIAM Journal on Computing*, 47(5):1888–1938, 2018.

[6] Census Bureau. Census disclosure avoidance system, 2020. `https://www.census.gov/programs-surveys/decennial-census/decade/2020/planning-management/process/disclosure-avoidance.html`.

[7] Adrian Rivera Cardoso and Ryan Rogers. Differentially private histograms under continual observation: Streaming selection into the unknown. *CoRR*, abs/2103.16787, 2021.

[8] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *IACR Cryptol. ePrint Arch.*, 2010:76, 2010.

[9] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *International Conference on the Theory and Applications of Cryptographic Techniques*, EUROCRYPT '06, pages 486–503, St. Petersburg, Russia, 2006.

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

[11] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 715–724. ACM, 2010.

[12] Cynthia Dwork, Moni Naor, Toniann Pitassi, Guy N. Rothblum, and Sergey Yekhanin. Pan-private streaming algorithms. In Andrew Chi-Chih Yao, editor, *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 66–80. Tsinghua University Press, 2010.

[13] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 51–60. IEEE Computer Society, 2010.

[14] Hendrik Fichtenberger, Monika Henzinger, and Wolfgang Ost. Differentially private algorithms for graphs under continual observation. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, pages 42:1–42:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[15] Abhradeep Guha Thakurta and Adam Smith. (Nearly) optimal algorithms for private online learning in full-information and bandit settings. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[16] Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2348–2356, 2012.

[17] Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In *Proceedings of the 42nd Annual ACM Symposium on the Theory of Computing*, STOC '10, pages 705–714, New York, NY, USA, 2010. ACM.

[18] Avinatan Hassidim, Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. Adversarially robust streaming algorithms via differential privacy. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[19] Prateek Jain, Pravesh Kothari, and Abhradeep Thakurta. Differentially private online learning. In Shie Mannor, Nathan Srebro, and Robert C. Williamson, editors, *Proceedings of the 25th Annual Conference on Learning Theory*, volume 23 of *Proceedings of Machine Learning Research*, pages 24.1–24.34, Edinburgh, Scotland, 25–27 Jun 2012. JMLR Workshop and Conference Proceedings.

[20] Haim Kaplan, Yishay Mansour, Kobbi Nissim, and Uri Stemmer. Separating adaptive streaming from oblivious streaming. *CoRR*, abs/2101.10836, 2021.

[21] Ryan McKenna and Daniel R Sheldon. Permute-and-flip: A new mechanism for differentially private selection. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 193–203. Curran Associates, Inc., 2020.

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

[23] Victor Perrier, Hassan Jameel Asghar, and Dali Kaafar. Private continual release of real-valued data streams. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.

[24] Alfred Rényi. On measures of entropy and information. *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics, pages 547–561, Berkeley, Calif., 1961. University of California Press*, abs/2101.10836, 1961.

[25] Shuang Song, Susan Little, Sanjay Mehta, Staal A. Vinterbo, and Kamalika Chaudhuri. Differentially private continual release of graph statistics. *CoRR*, abs/1809.02575, 2018.

[26] Thomas Steinke and Jonathan R. Ullman. Tight lower bounds for differentially private selection. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 552–563. IEEE Computer Society, 2017.

[27] Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Nearly optimal private LASSO. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3025–3033, 2015.

[28] Jonathan Ullman, 2021. Personal communication.

A   Useful Concentration Inequalities

Lemma A.1. *For all random variables $R \sim \mathcal{N}(0, \sigma^2)$,*

$$\Pr[|R| > \ell] \leq 2e^{-\frac{\ell^2}{2\sigma^2}}.$$

Lemma A.2. *Consider m random variables $R_1, \ldots, R_m \sim \mathcal{N}(0, \sigma^2)$. Then*

$$\Pr\left[\max_{j \in [m]} |R_j| > \ell\right] \leq 2me^{-\frac{\ell^2}{2\sigma^2}}.$$

Proof. By a union bound and Lemma A.1,

$$\Pr\left[\max_{i \in [m]} |R_i| > \ell\right] = \Pr(\exists i \in [m] \text{ such that } |R_i| > \ell)$$

$$\leq \sum_{i=1}^{m} \Pr(|R_i| > \ell) \leq \sum_{i=1}^{m} 2e^{-\frac{\ell^2}{2\sigma^2}} = 2me^{-\frac{\ell^2}{2\sigma^2}}.$$

$\square$

A similar union bound argument yields the following concentration inequality on the maximum of the absolute values of i.i.d. Laplace random variables.

Lemma A.3. *Fix $m \in \mathbb{N}$, $\lambda > 0$. Consider m random variables $R_1, \ldots, R_m \sim Lap(\lambda)$. Then for all $a > 0$,*

$$\Pr\left(\max_{i \in [m]} |R_i| > \lambda(\log m + \log a)\right) \leq e^{-a}.$$