The Price of Differential Privacy under Continual Observation

Palak Jain *1 Sofya Raskhodnikova *1 Satchit Sivakumar *1 Adam Smith *1

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 output that is accurate for all the inputs received so far. We provide the first strong lower bounds on the error of continual release mechanisms. In particular, for two fundamental problems that are closely related to empirical risk minimization and widely studied and used in the standard (batch) model, we prove that the worst case error of every continual release algorithm is $\Omega(T^{1/3})$ times larger than that of the best batch algorithm. Previous work shows only a $\Omega(\log T)$ gap between the worst case error achievable in these two models. We also formulate a model that allows for adaptively selected inputs, thus capturing dependencies that arise in many applications of continual release. Even though, in general, both privacy and accuracy are harder to attain in this model, we show that our lower bounds are matched by the error of simple algorithms that work even for adaptively selected inputs.

1. Introduction
Differentially private (DP) data analysis (Dwork, McSherry, Nissim, and Smith, 2006b) studies the design of algorithms that publish aggregate statistics about input datasets while preserving the privacy of individuals whose data they contain. The published aggregates often include COVID-19 dashboards that display statistics about sensitive data are collected over time, and published models and statistics need to be updated regularly. Examples include COVID-19 dashboards that display statistics about the number of COVID cases and deaths, ad campaign analytics, recommendation systems, and predictive language models. To investigate privacy in these situations, Dwork, Naor, Pitassi, and Rothblum (2010a) and Chan, Shi, and Song (2011) introduced the continual release model (sometimes referred to as the continual observation model). In the continual release 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 DP if releasing the entire vector of $T$ outputs satisfies differential privacy. Some deployments of differential privacy already fit this model (Apple, 2017). Furthermore, the continual release model arises as an intermediate step inside the analysis of some batch algorithms, such as the DP-FTRL learning algorithm (Kairouz et al., 2021). Despite the model’s prevalence, the theoretical understanding of continual release is still limited, and only a handful of techniques have been developed to tackle it. The main challenge for privacy in this model is that each individual record contributes to outputs at multiple time steps.

Dwork et al. (2010a) and Chan et al. (2011) considered the problem of computing summation in the continual release model when each record consists of one bit. Dwork et al. (2010a) showed that an error of $\Omega(\log T)$ is necessary to privately release all running sums. Additionally, both works designed the binary tree mechanism, a continual release mechanism that achieves (additive) error $O(\log^2 T)$ for this problem. Since then, this mechanism has been shown to accurately solve (with polylog $T$ error) many problems in the continual release model. (Further related work is discussed in Appendix B.) Given this success, one might conjecture that these results could be extended to a wide range of problems or at least to problems closely related to summation. Indeed, the largest previously known gap between the worst case error achievable in the batch and continual release models is $\Omega(\log T)$, ex-

---

*Equal contribution 1Department of Computer Science, Boston University, Boston, Massachusetts, USA. Correspondence to: Palak Jain <palakj@bu.edu>, Sofya Raskhodnikova <sofya@bu.edu>, Adam Smith <ads22@bu.edu>, Satchit Sivakumar <satchit@bu.edu>.

hibited by summation.

1.1. Our Contributions

We ask what price DP algorithms must pay in accuracy to solve a problem in the continual release model instead of the batch model. We show that for two fundamental problems related to summation and widely studied in the batch model, the gap is exponentially larger than the log separation for summation. In addition, we formalize a more realistic and broadly applicable model of continual release that allows for adaptively chosen inputs. Surprisingly, for the problems we consider, there is no increase in the error when we step up our privacy and accuracy requirements to deal with this more challenging setting.

In the first problem, called MaxSum, each input consists of \(d\) binary attributes\(^1\) and the goal is to approximate the maximum of the attribute sums. We define the error of a continual-release 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 for the index returned by the mechanism at that time step.

Motivation Both problems are abstractions of practically relevant tasks. For instance, consider a company monitoring the performance of \(d\) predictive models on a sequence of labeled examples: if for each data point we record whether each model made a successful prediction, then MaxSum corresponds to the success rate of the best model, and SumSelect corresponds to empirical risk minimization—that is, the index of the best model. If the data collected by a public health agency consists of records indicating which of \(d\) medical conditions each person suffers from, then MaxSum corresponds to the number of cases of the most prevalent condition so far, and SumSelect corresponds to the name of this condition.

Algorithms for both tasks are key ingredients in differentially private solutions to more complex problems such as empirical risk minimization (Bassily et al., 2014), synthetic data generation (Hardt et al., 2012), and high-dimensional optimization (Talwar et al., 2015). Consequently, these tasks and their variants have been thoroughly investigated in several models. Known algorithms and lower bounds for these problems provide pivotal pieces of our current understanding of the central model (McSherry & Talwar, 2007).

Bafna & Ullman, 2017; Steinke & Ullman, 2017; Durfee & Rogers, 2019; Qiao et al., 2021), the local model (Kasiviswanathan et al., 2011; Duchi et al., 2013; Edmonds et al., 2020), the shuffle model, and the pan-private model (Cheu & Ullman, 2021). Additionally, accurate continual-release variants of SumSelect, if they existed, would enable variants of the DP-FTRL learning framework (Kairouz et al., 2021) for a wider range of parameter spaces than are currently known.

We prove tight bounds on the error for these two problems in the continual release model in terms of the stream length (or “time horizon”) \(T\), the number of attributes (or the dimension) \(d\), and the privacy parameter \(\varepsilon\). To provide a comparison to the continual release model, we assume that algorithms in the batch model get input datasets of size \(T\). Intuitively, a batch algorithm \(A\) is differentially private if, for all datasets \(x\) and \(x’\) that differ in one record, all events under the distributions \(A(x)\) and \(A(x’)\) have similar probabilities. The definition of differential privacy (Definition 2.2) takes two parameters: \(\varepsilon\) and \(\delta\). The setting where \(\delta = 0\) is referred to as pure differential privacy. To provide a meaningful privacy guarantee in the setting where \(\delta > 0\) (referred to as approximate differential privacy, the parameter \(\delta\) has to be small (Kasiviswanathan & Smith, 2014); in our case, \(\delta = o(\varepsilon/T^2)\)). 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 Dwork et al. (2010a) for discussion of these two variants.

Separation Between the Continual Release and the Batch Models We demonstrate a strong separation between the continual release and the batch models. A comparison of the error achievable in the two models is presented in Table 1. The first row gives results on Summation from previous work; the second and the third row give results on MaxSum and SumSelect. The first column summarizes the error in the batch model: \(O(1)\) for MaxSum and \(O(\log d)\) for SumSelect. The former is obtained by an instantiation of the Laplace mechanism of Dwork et al. (2006b) and the latter, by an instantiation of the exponential mechanism of McSherry & Talwar (2007). In contrast, we show that in the continual release model, these tasks require error that is polynomial in either \(T\) or \(d\); this is presented in the second column of the table (the results shown in this column are for approximate differential privacy).

More detailed versions of our results are given in Table 2. For approximate differential privacy, we show that when \(d\) is sufficiently large, MaxSum\(_d\) and SumSelect\(_d\) require error that scales as roughly \(\tilde{\Omega}(\sqrt{T})\). For pure differential privacy, the error required for MaxSum and SumSelect is even larger, roughly \(\Omega(\sqrt{T})\).

\(^{1}\)Our algorithms and analyses work more generally, when inputs are from \([0, 1]^d\) (not just \(\{0, 1\}^d\)).
Table 1. Bounds on the error of $(\epsilon, \delta)$-DP mechanisms in the continual release model compared to their batch model counterparts. 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 $\delta = o(\epsilon/T^2)$, and the upper bounds apply when $\delta > \text{poly}(\frac{1}{T})$. For simplicity, the dependence on $\epsilon$ is suppressed in the table.

<table>
<thead>
<tr>
<th>Batch Model</th>
<th>Continual Release Model</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summation</td>
<td>$\Theta(1)$</td>
<td>$O(\log T)$</td>
</tr>
<tr>
<td></td>
<td>$\Omega(\log T)$</td>
<td>$O(\log^2 T)$</td>
</tr>
<tr>
<td>MaxSum</td>
<td>$\Theta(1)$</td>
<td>$\tilde{\Omega} \left( \min \left{ \sqrt{T}, \sqrt{d} \right} \right)$</td>
</tr>
<tr>
<td></td>
<td>$\tilde{\Omega} \left( \min \left{ \sqrt{T}, \sqrt{d} \text{ polylog}(T) \right} \right)$</td>
<td>Cor. E.3, E.6</td>
</tr>
<tr>
<td>SumSelect</td>
<td>$\Theta(\log d)$</td>
<td>$\tilde{\Omega} \left( \min \left{ \sqrt{T \log^2 d}, \sqrt{d}, T \right} \right)$</td>
</tr>
<tr>
<td></td>
<td>$\tilde{O} \left( \min \left{ \sqrt{T \log^2 d}, \sqrt{d} \text{ polylog}(T), T \right} \right)$</td>
<td>Cor. E.3, E.6</td>
</tr>
</tbody>
</table>

Table 2. More detailed statements of our results on the additive error of $(\epsilon, 0)$-DP and $(\epsilon, \delta)$-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 $\delta = o(\epsilon/T^2)$, and the upper bounds apply when $\delta > \text{poly}(\frac{1}{T})$. All our results are stated and proved for $\epsilon < 1$, but they apply for $\epsilon$ bounded by any larger constant as well.

<table>
<thead>
<tr>
<th>Approximate DP ($\delta &gt; 0$)</th>
<th>Pure DP ($\delta = 0$)</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxSum</td>
<td>$\Omega \left( \min \left{ \sqrt{T \epsilon}, \sqrt{d \epsilon}, T \right} \right)$</td>
<td>$\tilde{\Omega} \left( \min \left{ \sqrt{T \epsilon}, \frac{d \epsilon}{T}, T \right} \right)$</td>
</tr>
<tr>
<td></td>
<td>$\tilde{O} \left( \min \left{ \sqrt{T \epsilon \text{ polylog}(T)}, T \right} \right)$</td>
<td>$\tilde{O} \left( \min \left{ \sqrt{T \epsilon \text{ polylog}(T)}, T \right} \right)$</td>
</tr>
<tr>
<td>SumSelect</td>
<td>$\tilde{\Omega} \left( \min \left{ \sqrt{T \epsilon \log^2 T}, \sqrt{d \epsilon}, T \right} \right)$</td>
<td>$\tilde{\Omega} \left( \min \left{ \sqrt{T \epsilon \log^2 T}, \frac{d \epsilon}{T}, T \right} \right)$</td>
</tr>
<tr>
<td></td>
<td>$\tilde{O} \left( \min \left{ \sqrt{T \epsilon \log^2 T \text{ polylog}(T)}, T \right} \right)$</td>
<td>$\tilde{O} \left( \min \left{ \sqrt{T \epsilon \log^2 T \text{ polylog}(T)}, T \right} \right)$</td>
</tr>
</tbody>
</table>

**Lower Bound Technique: Reductions via Sequential Embedding** We obtain our lower bounds via a novel sequential embedding technique. This approach embeds multiple separate instances of an appropriately chosen base problem on the same sensitive dataset in the batch model into a single instance of a continual release problem. It allows us to use a continual release algorithm to solve multiple instances of the base problem. We can then invoke lower bounds for the batch model to obtain hardness results in the continual release model. For both MaxSum and SumSelect, we can adjust the parameters of our reductions to get nearly tight lower bounds for all values of the dimension $d$ and time horizon $T$ for both pure and $(\epsilon, \delta)$-differential privacy.

For MaxSum, the corresponding base problem is to compute one of the sums over which the maximum is taken. Such sums correspond to 1-way marginals of a dataset, and we apply lower bounds for releasing all 1-way marginals in the batch model by Hardt & Talwar (2010) (for pure differential privacy) and Bun et al. (2018) (for $(\epsilon, \delta)$-differential privacy) in conjunction with our reduction. For SumSelect, the base task in the batch model is selecting the index of the largest coordinate sum restricted to a subset of coordinates.

Multiple disjoint instances of the base task are captured by a problem we call $k$-IndSelect$_d$, where $k$ indicates the number of instances. To obtain lower bounds for this problem in the batch model, we use a simple packing argument in the case of pure differential privacy and prove a new lower bound for selecting top-$k$ sums using a result of Steinke & Ullman (2017) for a related problem in the case of $(\epsilon, \delta)$-differential privacy.

The main idea behind our sequential embedding technique is to embed a sensitive dataset into the first part of the stream given to the continual release algorithm. The remaining parts of the stream do not depend on the dataset and are chosen to force the continual release algorithm to output good approximations to separate instances of the base problem on the sensitive dataset. The technique was subsequently used to obtain lower bounds for other problems (Ghazi et al., 2023); see Section 1.2.

**Continual Release with Adaptively Chosen Inputs** The continual release model of Dwork et al. (2010a) and Chan et al. (2011) assumes that the input stream is chosen obliviously, before the algorithm is run. This means that the data at time $t$ cannot depend on the values the algorithm
Privacy analysis in the adaptive setting is subtle. In the
ity 1 in Appendix E.

continual release model with adaptively chosen inputs for
of the noisy values. We analyze these mechanisms in the
SumSelect takes the maximum (or, in the case of
argmax)
coordinates separately and

the value of the desired statistic (e.g.,
MaxSum)
across all parameter regimes.

In general, both privacy and accuracy are harder to attain
in this model\footnote{In a subsequent work, Denisov et al. (2022) give an example of a protocol that is private in the original continual release model, but not in the model with adaptively chosen inputs.}. However, for the specific problems we consider, it turns out that there is no overhead in terms of accuracy when the input stream is selected adaptively. We show that our lower bounds (that hold even when the input stream is selected obliviously) are matched by algorithms that work even with adaptively selected streams. We achieve this by analyzing variants of classical continual release algorithms in the new model.

Each of our lower bounds (summarized in Table 2) is the minimum of three terms, corresponding to different parameter regimes. They are matched (up to polylogarithmic factors in $T$ and $1/\delta$), in each regime, by the best of 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 SumSelect, argmax) of the noisy values. We analyze these mechanisms in the continual release model with adaptively chosen inputs for MaxSum, SumSelect, and general functions of sensitivity 1 in Appendix E.

Privacy analysis in the adaptive setting is subtle. In the nonadaptive setting, one argues that the algorithm’s outputs on any two fixed datasets that differ in exactly one element are indistinguishable. In contrast, the two input streams in the adaptive setting may diverge in an arbitrary number of records based on prior outputs of the mechanism. Because of this, one cannot generally reduce the proof of privacy with adaptively chosen inputs to a proof in the nonadaptive model. Instead, we build on techniques from simulation-based proofs in cryptography to argue directly, for specific algorithms, that the joint distributions of the two input streams and their corresponding output distributions are indistinguishable.

We note that adaptive composition—a standard tool in the analysis of differentially private mechanisms—is inadequate for dealing with the issue of adaptively chosen inputs. A continual release mechanism does not have to use independent randomness at each time step so one cannot, in general, apply adaptive composition directly.

1.2. Discussion and Open Questions

This paper has two key contributions: first, we establish strong lower bounds on the error of continual release mechanisms; second, we introduce the model with adaptively chosen inputs and then analyze algorithms for MaxSum and SumSelect in this model. Together, our mechanisms and lower bounds provide a comprehensive characterization of the error for MaxSum and SumSelect up to polylogarithmic factors in $T$ and $1/\delta$ across all parameter regimes.

Our new model is relevant in practical settings: our analysis applies to a deployed (binary-tree based) machine learning protocol by Kairouz et al. (2021). Furthermore, a follow-up on our work by Denisov et al. (2022) investigates improvements to the binary tree mechanism that have been deployed and proves they are private in the model with adaptively chosen inputs. The structure of the mechanisms they consider requires the use of the adaptive model in the analysis even when the data are fixed ahead of time.

In the course of their work, Denisov et al. (2022) delve deeper into the model with adaptively chosen inputs: they show that any mechanism that adds additive Gaussian noise is private in the adaptive model; they also construct an (artificial) protocol that is DP in the continual release model with nonadaptive inputs, but not DP with adaptive inputs. It is open to separate the two continual release models in the sense our work separates the batch model and the continual release model: by providing problems that require a large error blowup in the more demanding model.

Our lower bounds point to fundamental differences between the continual release and the batch models. In the batch model, low sensitivity of a function can be easily exploited to provide an accurate DP algorithm for releasing
The Price of Differential Privacy under Continual Observation

this function. It was consistent with prior work that a version of the widely-used exponential mechanism tailored to SumSelect in the continual release model could match the accuracy of summation. Our work rules out this possibility. We show that in the continual release model, low sensitivity alone is insufficient. To get DP mechanisms with very low (say, polylogarthmic in \( T \)) error for problems that do not reduce to low-dimensional summation or histograms, one must find and exploit new kinds of structure in the function being repeatedly evaluated.

Notably, our lower bounds apply even to “offline” algorithms that receive the entire input stream before producing output; i.e., they do not rely on the algorithm’s uncertainty of what comes later in the stream. It would be interesting to find natural problems that separate the offline and the continual release models, as discussed in Denisov et al. (2022).

Finally, our sequential embedding technique has already found application in follow-up work of Ghazi et al. (2023), where it is used to establish lower bounds for other practically important problems such as counting distinct elements.

1.3. Organization of Paper

In Section 2, we define the continual release model with nonadaptively chosen inputs and state the problems we consider. Additional preliminaries appear in Appendix A. Related work not covered in the introduction is described in Appendix B. Sections 3–5 present our technical results and the definition of the continual release model with adaptively chosen inputs. All omitted proofs appear in appendices.

2. Definitions

2.1. Continual Release with Nonadaptively Chosen Inputs

A mechanism in the continual release model (Dwork et al., 2010a; Chan et al., 2011) is an algorithm that receives its input \( x = (x_1, \ldots, x_T) \in X^T \) 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 \( \alpha \). We use \( x_{[t]} = (x_1, \ldots, x_t) \) for \( t \in [T] \) to denote the first \( t \) records in a stream \( x \) (similarly, \( a_{[t]} = (a_1, \ldots, a_t) \)). 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. (1) The continual release model of Dwork et al. (2010a) and Chan et al. (2011): This model assumes that the input stream \( x \) is fixed before the mechanism runs. This means that the data at time \( t \) cannot depend on the values the algorithm returned at prior steps, even though they arrive after earlier outputs are released. (2) The continual release model with adaptively chosen inputs: This model 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. This model gives a more faithful representation of real-life settings, where the data may change based on prior feedback from the algorithm. We formalize this model in Section 5.1.

All our lower bounds are for the model with nonadaptively chosen inputs and, consequently, imply the same lower bounds for the model with adaptively chosen inputs. In contrast, all our algorithms work with adaptively chosen inputs (and, consequently, in the special case when inputs are chosen nonadaptively).

We refer to standard algorithms that get their input in one batch and produce one output as \( \text{batch} \) algorithms. For clarity, we refer to continual release algorithms as \( \text{mechanisms} \).

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 \( f : X^* \to Y \), a dataset \( x \in X^* \), and an answer \( a \in Y \), let \( \text{ERR}_f(x, a) \) be a nonnegative number that quantifies how far off \( a \) is from \( f(x) \). Specifically, when \( Y = \mathbb{R}^k \),

\[
\text{ERR}_f(x, a) = \|f(x) - a\|_\infty.
\]

Later (in (2)), we define a different notion of error for the optimization problem SumSelect. The error for an optimization problem corresponds to the deficit in the objective function.

Definition 2.1 (Accuracy of a Mechanism). In the continual release model with nonadaptively chosen inputs, a mechanism \( M \) is \((\alpha, T)\)-accurate for \( f \) if, for all fixed input streams \( x = (x_1, \ldots, x_T) \), the maximum error \( \text{ERR}_f(x_{[t]}, a_{[t]}) \) over the outputs \( a_1, \ldots, a_T \) of mechanism \( M \) is bounded by \( \alpha \) with high probability, that is,

\[
\Pr_{\text{coins of } M} \left[ \max_{t \in [T]} \text{ERR}_f(x_{[t]}, a_{[t]}) \leq \alpha \right] \geq \frac{2}{3}.
\]

Privacy Finally, we define privacy in the continual release model with nonadaptively chosen inputs.

Definition 2.2 (Privacy of a Mechanism). Given a mechanism \( M \), define \( A_M \) to be the batch model algorithm that receives an input dataset \( x \), runs \( M \) on stream \( x \), and returns the output stream \( \alpha \) of \( M \). The mechanism \( M \) is \((\varepsilon, \delta)\)-differentially private (DP) in the continual release model with nonadaptively chosen inputs if \( A_M \) is \((\varepsilon, \delta)\)-DP in the batch model.

Definition 2.2 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.

2.2. Problem Definitions

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

**Definition 2.3.** Let $d \in \mathbb{N}$ and $X = \{0, 1\}^d$. For a dataset $x \in X^*$ and $j \in [d]$, the $j^{th}$ attribute of record $x_i$ is its $j^{th}$ coordinate, denoted $x_i[j]$. Let $t \in \mathbb{N}$ and $x_t \in X^t$. The function $\text{MaxSum}_d: X^* \rightarrow \mathbb{N}$ is

$$\text{MaxSum}_d(x_t) \overset{\text{def}}{=} \max_{j \in [d]} \left( \sum_{i \in [t]} x_i[j] \right).$$

The function $\text{SumSelect}_d: X^* \rightarrow [d]$ is

$$\text{SumSelect}_d(x_t) \overset{\text{def}}{=} \arg \max_{j \in [d]} \left( \sum_{i \in [t]} x_i[j] \right).$$

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 2.1, uses the notion $\text{ERR}_f$. We define the error $\text{ERR}_{\text{MaxSum}}$ as in (1). For $\text{SumSelect}$, it is defined by:

$$\text{ERR}_{\text{SumSelect}}(x_t, a_t) = \text{MaxSum}_d(x_t) - \sum_{i \in [t]} x_i[a_i].$$

(2)

3. Lower Bounds for $\text{MaxSum}$

In this section, we prove Theorem 3.1 that provides strong lower bounds on the accuracy parameter $\alpha$ for any accurate mechanism for $\text{MaxSum}_d$ in the continual release model with nonadaptively chosen inputs. Our lower bounds match the upper bounds from Section 5 for $\text{MaxSum}_d$ in the continual release model with adaptively chosen inputs 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 continual release model with nonadaptively chosen inputs that are $(\varepsilon, \delta)$-differentially private and $(\alpha, T)$-accurate for $\text{MaxSum}_d$, the following statements hold.

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

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

$\text{MaxSum}_d$ can be released in the batch model with $\alpha = O(1/\varepsilon)$ via the Laplace mechanism (Dwork et al., 2006b). Hence, Theorem 3.1 shows a strong separation between the batch model of differential privacy and continual release.

3.1. 1-way Marginal Queries in Batch Model

To prove our lower bounds for $\text{MaxSum}$, we reduce from the problem of approximating 1-way marginals in the batch model. The function $\text{Marginals}_d : X^* \rightarrow [0, 1]^d$ maps a dataset $y$ of any size $n$ to a vector $(q_1(y), \ldots, q_d(y))$, where $q_j$, called the $j^{th}$ marginal, is defined as $q_j(y) = \frac{1}{n} \sum_{i=1}^n y_i[j]$. The error $\text{ERR}_{\text{Marginals}}$ is defined as in (1). 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 $X = \{0, 1\}^d$. Let $f : X^n \rightarrow \mathbb{R}^d$ be a function on datasets. Batch algorithm $\mathcal{A}$ is $(\gamma, n)$-accurate for $f$ if for all datasets $y \in X^n$,

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

We use the lower bounds from Bun et al. (2018); Hardt & Talwar (2010) for the problem of estimating $\text{Marginals}_d$ 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 Hardt & Talwar (2010) 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 $(\varepsilon, \delta)$-differentially private and $(\gamma, n)$-accurate for $\text{Marginals}_d$, the following statements hold.

1. Bun et al. (2018): If $\delta > 0$ and $\delta = o(1/n)$, then
   $$n = \Omega \left( \frac{\sqrt{d}}{\varepsilon^{3/2} \log^{1/2}(1/\varepsilon T)} \right).$$

2. Hardt & Talwar (2010): If $\delta = 0$, then $n = \Omega \left( \frac{d}{\varepsilon^2} \right)$.

3.2. Proof Sketch of Theorem 3.1

We give a proof sketch of Theorem 3.1; formal details can be found in Appendix C. Let $\mathcal{M}$ be an $(\varepsilon, \delta)$-DP and $(\alpha, T)$-accurate mechanism for $\text{MaxSum}_d$ in the continual release model with nonadaptively chosen inputs. We use $\mathcal{M}$ to construct an $(\varepsilon, \delta)$-DP batch algorithm $\mathcal{A}$ that is $(\frac{\gamma}{n}, n)$-accurate for $\text{Marginals}_d$. The main idea in the construction, (presented in Algorithm 2 in Appendix C), 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, $\mathcal{A}$ sends its own dataset $y$ 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.

This gives an accurate algorithm for the marginals problem, which is captured in the following lemma.

**Lemma 3.4 (Informal).** Let \( \mathcal{A} \) be the algorithm informally described in the previous paragraph. 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 \( (\varepsilon, \delta) \)-DP and \( (\alpha, T) \)-accurate for \( \text{MaxSum}_d \) in the continual release model with nonadaptively chosen inputs, then batch algorithm \( \mathcal{A} \) is \( (\varepsilon, \delta) \)-DP and \( (\frac{\alpha}{n}, n) \)-accurate for \( \text{Marginals}_d \).

Observe that both lower bounds on \( \alpha \) 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 \( \alpha \).

The rest of the proof of Theorem 3.1 follows by a case analysis: (1) For \( \varepsilon \leq \frac{\alpha}{T} \), we use a group privacy argument to show that \( \alpha > T/9 \) (for both pure and approximate differential privacy). (2) For \( \varepsilon > \frac{\alpha}{T} \), we use Lemma 3.4 and Lemma 3.3 to lower bound \( \alpha \). The details of the proof can be found in Appendix C.

### 4. Lower Bounds for SumSelect

In this section, we prove Theorem 4.1 that provides strong lower bounds on the accuracy parameter \( \alpha \) of any \( (\alpha, T) \)-accurate algorithm \( \mathcal{M} \) for \( \text{SumSelect}_d \) in the continual release model with nonadaptively chosen inputs. Our lower bounds match the upper bounds from Section 5 for \( \text{SumSelect}_d \) in the continual release model with adaptively chosen inputs up to logarithmic factors in the time horizon \( T \) and the number of coordinates \( d \). We give the formal details of the proof in Appendix D.

**Theorem 4.1.** For all \( \varepsilon \in (0, 1], \delta \in [0, 1), \alpha > 0 \), sufficiently large \( d, T \in \mathbb{N} \), and mechanisms \( \mathcal{M} \) in the continual release model with nonadaptively chosen inputs that are \( (\varepsilon, \delta) \)-DP and \( (\alpha, T) \)-accurate for \( \text{SumSelect}_d \), the following statements hold.

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

### 4.1. Proof sketch of Theorem 4.1

To prove our lower bounds for \( \text{SumSelect} \) in the continual release model with nonadaptively chosen inputs, we first reduce from a problem called \( k \)-IndSelect in the batch model. In the batch model, \( k \)-IndSelect solves the problem of selecting the index of the largest coordinate sum in each of \( k \) disjoint subsets (which have \( d \) coordinates each). We reduce the problem of solving \( k \)-IndSelect in the batch model to solving \( \text{SumSelect}_{dk} \) in the continual release model.

We then prove new lower bounds for \( k \)-IndSelect in the batch model: (1) In the case of approximate differential privacy \( (\varepsilon, \delta) \)-DP, we obtain a lower bound for \( k \)-IndSelect by reducing to a related problem and invoking a result of Steinke & Ullman (2017). (2) In the case of pure differential privacy \( (\varepsilon, 0) \)-DP, we prove a lower bound by using a standard packing argument.

Finally, we use the new lower bounds for \( k \)-IndSelect in conjunction with our reduction and careful case analysis to obtain our lower bounds in the continual release model.

### 5. Continual Release with Adaptively Chosen Inputs

In this section, we provide an explicit game-based formulation of the continual release model with adaptively chosen inputs. Here, the inputs to the mechanism can be chosen online by an adversary that observes the outputs of the mechanism on prior stream elements. Unlike the setting considered in the prior sections, the input data at time \( t \) in this setting can depend on the values returned by the algorithm at prior steps. Because of this, both privacy and accuracy are harder to attain in this model in general.

Later in the section we describe differentially private mechanisms for two types of problems: \( \text{SumSelect} \) and approximating functions with bounded sensitivity \( (\ell_2 \text{ sensitivity in the case of approximate differential privacy and } \ell_1 \text{ sensitivity in the case of pure DP}) \). For these problems we show that there is no overhead in terms of accuracy when the input stream is selected adaptively, that is, our lower bounds (that hold even when the input stream is selected obliviously) are matched by algorithms that work even with adaptively selected streams. We achieve this by analyzing variants of classical continual release algorithms in the new model. Since a proof of privacy in the model with adaptively chosen inputs cannot be generally reduced to a

\text{MaxSum}, which has sensitivity 1.
proof in the nonadaptive model, our analysis adapts techniques from simulation-based cryptography to argue indistinguishability of the relevant distributions directly. The main challenge in analyzing privacy is that input streams in ‘neighboring’ interactions with the private mechanism may differ in many records, which necessitates using different techniques than privacy proofs in prior work, which analyze privacy with respect to input streams that differ in a single record.

Our mechanisms are \((\alpha, T)\)-accurate, where the upper bounds for \(\alpha\) match the lower bounds obtained in previous sections in the continual release model with nonadaptively chosen inputs up to logarithmic factors in the time horizon \(T\), the number of coordinates \(d\), and the inverse of the privacy parameter \(\frac{1}{\alpha}\).

### 5.1. Model Definition

In the continual release model with adaptively chosen inputs, a mechanism \(M\) interacts with a randomized adversarial process \(\text{Adv}\) that runs for \(T\) timesteps; at timestep \(t \in [T]\), the process \(\text{Adv}\) receives \(a_t\) from \(M\), updates its internal state, and produces input record \(x_{t+1}\) that is sent to \(M\) at timestep \(t + 1\). The adversarial process \(\text{Adv}\) can choose \(x_{t+1}\) based on the previous input records \(x_t\) and \(M\)’s previous outputs \(a_t\). We make no assumptions on \(\text{Adv}\) regarding running time or complexity; its only limitation is that it does not see the internal coins of \(M\).

**Definition 5.1.** A mechanism \(M\) is \((\alpha, T)\)-accurate for a function \(f\) in the continual release model with adaptively chosen inputs if for all processes \(\text{Adv}\), the error of \(M\) with respect to \(\text{Adv}\) is at most \(\alpha\) with high probability, that is,

\[
\Pr_{\text{coins of } M, \text{Adv}}\left[\max_{t \in [T]} \text{ERR}_f(a_t; x_t) \leq \alpha\right] \geq \frac{2}{3}.
\]

A similar notion of accuracy was considered in work on adversarial streaming (Ben-Eliezer et al., 2020; Hassidim et al., 2022; Kaplan et al., 2021), though those articles do not directly address privacy.

Next, we define (event-level) privacy in the continual release model with adaptively chosen inputs which is trickier than in the continual release model with nonadaptively chosen inputs. One difficulty is that the definition of ‘neighboring’ input streams must still allow for adaptive online generation of the input streams by an adversarial process \(\text{Adv}\). This concept is implicit in the work of Thakurta & Smith (2013), but to our knowledge has not been previously defined. Privacy is defined with respect to the game \(\Pi_{M, \text{Adv}}\), described in Algorithm 1.

---

**Algorithm 1** Privacy game \(\Pi_{M, \text{Adv}}\) for the continual release model with adaptively chosen inputs

1. **Input:** time horizon \(T \in \mathbb{N}\), side \(\{L, R\}\) (not known to \(\text{Adv}\)).
2. **for** \(t = 1\) to \(T\) **do**
3. \(\text{Adv}\) outputs \(x_t \in \{\text{challenge, regular}\}\), where challenge is chosen once during the game.
4. **if** \(\text{type}_t = \text{regular then}\)
5. \(\text{Adv}\) outputs \(x_t \in \mathcal{X}\) which is sent to \(M\).
6. **end if**
7. **if** \(\text{type}_t = \text{challenge then}\)
8. \(t^* \leftarrow t\).
9. \(\text{Adv}\) outputs \((x_{t^*}^{(L)}, x_{t^*}^{(R)}) \in \mathcal{X}^2\).
10. \(x_t^{(\text{side})}\) is sent to \(M\).
11. **end if**
12. \(M\) outputs \(a_t\) which is given to \(\text{Adv}\).
13. **end for**

---

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

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 \((\varepsilon, \delta)\)-DP, we consider a related notion, called \(z\text{CDP}\) (Bun & Steinke, 2016). See Section A.1 for background on \(z\text{CDP}\) and the notion of \(\rho\)-closeness of random variables \((\simeq_{\rho})\).
Definition 5.3. A mechanism $\mathcal{M}$ is $(\varepsilon, \delta)$-DP in the continual release model with adaptively chosen inputs if, for all adversaries $\mathcal{A}$,

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

A mechanism $\mathcal{M}$ is $\rho$-zCDP in the continual release model with adaptively chosen inputs if for all adversaries $\mathcal{A}$,

$$V_{\mathcal{M}, \mathcal{A}_{\text{Adv}}}^{(L)} \approx_{\rho} V_{\mathcal{M}, \mathcal{A}_{\text{Adv}}}^{(R)}.$$ 

The symbol $\approx_{\rho}$ denotes $\rho$-closeness (Definition A.11).

5.2. Summary of Upper Bounds for Adaptive Inputs

Our upper bounds on the error of differentially private mechanisms for $\text{MaxSum}_d$ and $\text{SumSelect}_d$ in the continual release model with adaptively chosen inputs are summarized in Table 3. The corresponding theorems are stated in Appendix E.1. The upper bounds in the table are attained by two simple mechanisms: one uses the binary tree mechanism and the other recomputes the target function at regular intervals. The bounds stated for $\text{MaxSum}_d$ and obtained via recomputing periodically apply more generally: to all sensitivity-1 functions. Detailed proofs are given in Appendix E.

Table 3. Our (asymptotic) upper bounds on the error of continual release mechanisms with adaptively chosen inputs.

<table>
<thead>
<tr>
<th>Problem</th>
<th>$\rho$-zCDP</th>
<th>$(\varepsilon, 0)$-DP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{MaxSum}$</td>
<td>$\sqrt{d \log T \sqrt{T \log d(T)}}$</td>
<td>$\frac{d \log d}{\varepsilon} \log \frac{T \log T}{\varepsilon}$</td>
</tr>
<tr>
<td>$\text{SumSelect}$</td>
<td>$\sqrt{d \log T \sqrt{T \log d(T)}}$</td>
<td>$\frac{d \log d}{\varepsilon} \log \frac{T \log d(T)}{\varepsilon}$</td>
</tr>
</tbody>
</table>

Acknowledgments

We are grateful to Kobbi Nissim for being part of the conversations that got this work started and for subsequent helpful comments. We are also grateful to Jon Ullman and Thomas Steinke for insights into lower bounds for the top-$k$ selection problem. A.S. and P.J. were supported in part by NSF awards CCF-1763786 and CNS-2120667 as well as Faculty Awards from Google and Apple. S.S. was supported by NSF award CNS-2046425 and Cooperative Agreement CB20ADR0160001 with the Census Bureau.

References


The Price of Differential Privacy under Continual Observation


The Price of Differential Privacy under Continual Observation


A. Differential Privacy

We first introduce the notion of \((\varepsilon, \delta)\)-indistinguishability.

**Definition A.1** \(((\varepsilon, \delta)\)-Indistinguishability). Random variables \(R_1\) and \(R_2\) over the same outcome space \(\mathcal{Y}\) are \((\varepsilon, \delta)\)-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 \(x = (x_1, \ldots, x_n) \in \mathcal{X}^n\) 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 A.2** (Differential Privacy in Batch Model (Dwork et al., 2006b,a)). A randomized algorithm \(\mathcal{A} : \mathcal{X}^n \rightarrow \mathcal{Y}\) is \((\varepsilon, \delta)\)-differentially private (DP) if for every pair of neighboring datasets \(x, x' \in \mathcal{X}^n\),

\[
\mathcal{A}(x) \approx_{\varepsilon, \delta} \mathcal{A}(x').
\]

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 A.3** (Group Privacy (Dwork et al., 2006b)). Every \((\varepsilon, \delta)\)-DP algorithm \(\mathcal{A}\) is \((\ell \varepsilon, \delta')\)-DP for groups of size \(\ell\), where \(\delta' = \delta e^{\ell \varepsilon - 1}\); that is, for all datasets \(x, x'\) such that \(\|x - x'\|_0 \leq \ell\),

\[
\mathcal{A}(x) \approx_{\ell \varepsilon, \delta'} \mathcal{A}(x').
\]

Differential privacy is closed under post-processing.

**Lemma A.4** (Post-Processing (Dwork et al., 2006b; Bun & Steinke, 2016)). If \(\mathcal{A}\) is an \((\varepsilon, \delta)\)-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 \((\varepsilon, \delta)\)-DP.

**Definition A.5** (Sensitivity). Let \(f : \mathcal{X}^n \rightarrow \mathbb{R}^m\) be a function. Its \(\ell_1\)-sensitivity is

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

To define \(\ell_2\)-sensitivity, we replace the \(\ell_1\) norm with the \(\ell_2\) norm.

Our algorithms use the standard Laplace and Exponential mechanisms to ensure differential privacy.

**Definition A.6** (Laplace Distribution). The Laplace distribution with parameter \(b\) and mean 0, denoted \(\text{Lap}(b)\), has probability density

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

**Lemma A.7** (Laplace Mechanism, (Dwork et al., 2006b)). Let \(f : \mathcal{X}^n \rightarrow \mathbb{R}^m\) be a function with \(\ell_1\)-sensitivity at most \(\Delta\). Then the Laplace mechanism is algorithm

\[
\mathcal{A}_f(x) = f(x) + (Z_1, \ldots, Z_m),
\]

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

**Lemma A.8** (Exponential Mechanism (McSherry & Talwar, 2007)). Let \(L\) be a set of outputs and \(g : L \times \mathcal{X}^n \rightarrow \mathbb{R}\) be a function that measures the quality of each output on a dataset. Assume that for every \(m \in L\), the function \(g(m, \cdot)\) has \(\ell_1\)-sensitivity at most \(\Delta\). Then, for all \(\varepsilon > 0\) 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

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

**Definition A.9** (Gaussian Distribution). The Gaussian distribution with parameter \(\sigma\) and mean 0, denoted \(\mathcal{N}(0, \sigma^2)\), has probability density

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

A.1. \(\rho\)-zCDP

This section describes “zero-concentrated differential privacy” (zCDP), a variant of differential privacy that is less stringent than pure differential privacy, but more stringent than approximate differential privacy. In contrast to \((\varepsilon, \delta)\)-differential privacy, zCDP requires output distributions on all pairs of neighboring datasets to be \(\rho\)-close (Definition A.11) instead of \((\varepsilon, \delta)\)-indistinguishable. In Appendix E, we show that our algorithms are zCDP and then use conversion from zCDP to \((\varepsilon, \delta)\)-differential privacy (Lemma A.15) to restate our upper bounds in the same terms as our lower bounds for easy comparison between the two.

**Definition A.10** (Rényi Divergence (Rényi, 1961)). Let \(Q\) and \(Q'\) be distributions on \(\mathcal{Y}\). For \(\xi \in (1, \infty)\), the Rényi divergence of order \(\xi\) between \(Q\) and \(Q'\) (also called the \(\xi\)-Rényi Divergence) is defined as

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

(3)

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 \(Q(\cdot)\) with the the Radon-Nikodym derivative of \(Q\) with respect to \(Q'\).
Algorithm

Suppose $A$ (Composition (Bun & Steinke, 2016))

Lemma A.13

analysing the privacy of the algorithms in Appendix E.

One major benefit of using zCDP is that this definition of differential 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.

User-level differential privacy

User-level privacy in the continual release model was first studied by Dwork et al. (2010a) and Chan et al. (2011). 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. (2010b), 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 Dwork et al. (2010b) for details.

C. Proofs Omitted from Section 3

In this section, we prove Theorem 3.1 by formalizing the proof sketch from Section 3.2.

For vectors $u = (u_1, \ldots, u_\ell)$ and $v = (v_1, \ldots, v_m)$, let $u \circ v = (u_1, \ldots, u_\ell, v_1, \ldots, v_m)$. For a vector $v$, let $v^n$
Lemma A.4 implies that \( A_x M \) for \( 0 \leq d \leq 1 \), the attribute with the largest sum in \( A \) is \( \sum_{i \in \{2, \ldots, n\}} x_i[j] \). When it is run on \( y \otimes \cdots \otimes v \), whereas the maximum sum of any attribute in \( y \) is \( n \).

Algorithm 2: Algorithm \( \mathcal{A} \) for estimating all 1-way marginals

1. **Input:** \( y = (y_1, \ldots, y_n) \in \mathcal{X}^n \), where \( \mathcal{X} = \{0,1\}^d \), and black-box access to mechanism \( M \).
2. **Output:** \( b = (b_1, \ldots, b_d) \in \mathbb{R}^d \).
3. Let \( e_j \), be a vector of length \( d \) with 1 in coordinate \( j \) and 0 everywhere else; let \( \overline{e_j} \leftarrow (1 - e_j) \).
4. Construct a stream \( x \leftarrow y \circ (e_1)^n \circ (\overline{e_1})^n \circ \cdots \circ (e_{d-1})^n \circ (\overline{e_{d-1}})^n \circ e_d^n \) with \( 2dn \) records.
5. for \( t \in [T] \) do
6. Send \( x_t \) to \( M \) and get the corresponding output \( a_t \).
7. end for
8. for \( j \in [d] \) do
9. \( b_j \leftarrow a_{2jn}/n - j \).
10. end for
11. Output \( b \leftarrow (b_1, \ldots, b_d) \).

Since the transformation from \( M \) to \( \mathcal{A} \) is deterministic, the coins of \( \mathcal{A} \) are the same as the coins of \( M \). By (4) and the computation of the estimates for the Marginals in Step 9 of Algorithm 2.

\[
\begin{align*}
\Pr_{\text{coins of } \mathcal{A}} \left[ \text{ERR}_{\text{Marginals}}(y, \mathcal{A}(y)) \leq \frac{\alpha}{n} \right] &= \Pr_{\text{coins of } \mathcal{A}} \left[ \max_{j \in [d]} |q_j(y) - b_j| \leq \frac{\alpha}{n} \right] \\
&= \Pr_{\text{coins of } M} \left[ \max_{t \in \{2n, \ldots, 2dn\}} \left| \text{MaxSum}_d(x_{[t]}^n) - a_t \right| \leq \alpha \right] \\
&\geq \Pr_{\text{coins of } M} \left[ \max_{t \in [T]} \left| \text{MaxSum}_d(x_{[t]}^n) - a_t \right| \leq \alpha \right] \\
&= \Pr_{\text{coins of } M} \left[ \max_{t \in [T]} \text{ERR}_{\text{MaxSum}}(x_{[t]}^n, a_t) \leq \alpha \right] \geq \frac{2}{3},
\end{align*}
\]

where we used that \( M \) is \((\alpha, T)\)-accurate for \( \text{MaxSum}_d \). Thus, \( \mathcal{A} \) is \((\frac{\alpha}{n}, n)\)-accurate for Marginals.

Now, we are ready to prove Theorem 3.1.

**Proof of Theorem 3.1.** The accuracy parameter \( \alpha \) is non-decreasing as a function of \( d \) since any mechanism \( M \) for \( \text{MaxSum}_d \) can be used to approximate \( \text{MaxSum}_{d'} \) 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 \( \alpha \) 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 \( \alpha \).

First, consider the case when \( \varepsilon \leq \frac{\delta}{2} \). We will show that in this case (for both pure and approximate differential privacy), \( \alpha > T/9 \). Since \( \delta \) is a nondecreasing function of \( d \), it is sufficient to show this for \( d = 1 \). Suppose for the sake of contradiction that \( \alpha \leq T/9 \). Let \( x = (0)^T \) and \( x' = (0)^{(T\ln(1/\delta))/4}(1)^{T/4} \) be datastreams that differ on \( T/4 \) records. Let \( a_T \) and \( a_T' \) be the final outputs of \( M \) on input streams \( x \) and \( x' \), respectively. By accuracy of \( M \), we have \( \Pr[a_T \leq T/9] \leq \frac{2}{3} \). Applying Lemma A.3 on group privacy with \( \varepsilon' = \frac{T}{2} \) and \( \ell = T/4 \), we get \( \Pr[a_T' > T/9] \leq \sqrt{\frac{T}{4}} \cdot \Pr[a_T > T/9] + \frac{2\alpha}{T/9} < 2/3 \) for sufficiently large \( T \), since \( \delta = \alpha \varepsilon / T \). But \( \text{MaxSum}_d(x') = T/4 \), so \( M \) is not \( (T/9, T/4) \)-accurate, a contradiction. Hence, \( \alpha = \Omega(T) \).

Now assume \( \varepsilon > \frac{\delta}{2} \), i.e., \( \varepsilon T > 2 \). We start by proving Item 1 (when \( \delta = \alpha \varepsilon / T \)). Let \( \mathcal{A} \) be the algorithm for Marginals with black-box access to \( M \), as defined in Algorithm 2. If \( T \geq 2dn \) and \( \frac{\alpha}{n} < 1 \), then by Lemma C.1, algorithm \( \mathcal{A} \) is \((\varepsilon, \delta)\)-differentially private and \((\frac{\alpha}{n}, n)\)-accurate for Marginals. (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 \( \alpha \).
Case 1: \( d \leq (\varepsilon T \log(\varepsilon T))^{2/3} \). If there exists a dataset size \( n \in (\alpha, \frac{T}{e^2}) \), then by Item 1 of Lemma 3.3, \( n = \Omega \left( \frac{\sqrt{\varepsilon}}{\log \frac{\varepsilon}{T}} \right) \), and hence \( \alpha = \Omega \left( \frac{\sqrt{\varepsilon}}{\log \frac{\varepsilon}{T}} \right) \). If no such \( n \) exists, then \( n + 1 \geq \frac{T}{e^2} \), and hence \( \alpha = \Omega \left( \frac{T}{e^2} \right) = \Omega \left( \frac{T^{1/3}}{\varepsilon^{2/3} \log^{2/3}(\varepsilon T)} \right) \). 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 \( d' = \lceil (\varepsilon T \log(\varepsilon T))^{2/3} \rceil \). Observe that \( d' \geq 1 \) because \( \varepsilon T > 2 \). By our previous padding argument, a mechanism for \( \text{MaxSum}_d \) can be used to approximate \( \text{MaxSum}_{d'} \) for \( d' = (\varepsilon T)^{2/3} \) with the same accuracy and privacy guarantees. Therefore, \( \alpha = \Omega \left( \min \left\{ \frac{T^{1/3}}{\varepsilon^{2/3} \log^{2/3}(\varepsilon T)}, \frac{\sqrt{d}}{\varepsilon \log d} \right\} \right) \).

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 \( d < \sqrt{\varepsilon T} \) and \( d > \sqrt{\varepsilon T} \) and use Item 2 from Lemma 3.3 instead of Item 1. If a dataset size \( n \in (\alpha, \frac{T}{e^2}) \) exists, by Item 2 of Lemma 3.3, we get \( n = \Omega \left( \frac{n^2}{\alpha^2} \right) \), and hence \( \alpha = \Omega \left( \frac{n}{\alpha} \right) \). If no such \( n \) exists, then \( n + 1 \geq \frac{T}{e^2} \), and hence \( \alpha = \Omega \left( \frac{T}{d} \right) = \Omega \left( \frac{\sqrt{T}}{\varepsilon} \right) \). If \( d > \sqrt{\varepsilon T} \), a padding argument gives that \( \alpha = \Omega \left( \frac{\sqrt{T}}{\varepsilon} \right) \).

\( \square \)

D. Proofs Omitted from Section 4

To prove our lower bounds for \( \text{SumSelect} \) in the continual release model with nonadaptively chosen inputs, we reduce from a problem called \( k\text{-IndSelect} \) that solves \( k \) disjoint instances of the problem of selecting the index of the largest marginal in the batch model.

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

\[ h_r(\mathcal{Y}) = \text{SumSelect}_d \left( \mathcal{Y}[(r - 1)d + 1 : rd] \right). \]

The accuracy for \( k\text{-IndSelect} \) is defined as in Definition 3.2. To apply it, we define the error \( \text{ERR}_{k\text{-IndSelect}} \). Note that the error is scaled differently than for \( \text{SumSelect} \) 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 \( \text{ERR}_{k\text{-IndSelect}}(\mathbf{y}, \mathbf{b}) = \max_{r \in [k]} \left( \frac{1}{n} \cdot \text{ERR}_{\text{SumSelect}} \left( \mathcal{Y}[(r - 1)d + 1 : rd], b_r \right) \right) \).

Next, we give lower bounds for \((\varepsilon, \delta)\)-differentially private approximation of \( k\text{-IndSelect} \) in the batch model.

**Lemma D.1.** For all \( \varepsilon, \delta \in (0, 1), \gamma \in [0, \frac{1}{2}) \), sufficiently large \( d, k, n \in \mathbb{N} \), and batch algorithms \( \mathcal{A} \) that are \((\varepsilon, \delta)\)-differentially private and \((\gamma, n)\)-accurate for \( k\text{-IndSelect}_{\mathcal{A}} \), the following statements hold.

1. If \( \delta \in (0, \frac{1}{2\gamma}) \), then \( n = \Omega \left( \frac{\sqrt{\varepsilon \cdot \ln(1/\delta)}}{\log d} \right) \).

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

**Proof Sketch of Item 1 in Lemma D.1.** We are aware of two approaches to proving this result, both of which were communicated to us by Jonathan Ullman (Ullman, 2021). The first uses the top-\( k \) selection lower bound of Steinke & Ullman (2017). 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.

If one divides the coordinates into \( 10k \) 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 \( k\text{-IndSelect}_d \) 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 Steinke & Ullman (2017) on \( n \) then applies. The statement we give here relies on a strengthening of the main result in Steinke & Ullman (2017) that incorporates \( \delta \), communicated to us by Thomas Steinke.

Another approach is to use the composition framework of Bun et al. (2018). 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 Bun et al. (2018), one obtains a lower bound for \( k\text{-IndSelect}_d \).

\( \square \)

**Proof of Item 2 in Lemma D.1.** The proof proceeds via a standard packing argument. For \( \mathbf{u} \in [d]^k \), define \( \mathbf{y}^*_\mathbf{u} \in \{0, 1\}^{dk} \) 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 \( \mathbf{y}_\mathbf{u} \) be the dataset that consists of \( 2\gamma n \) copies of \( \mathbf{y}^*_\mathbf{u} \) 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 \((\gamma, n)\)-accurate, \( \Pr_{\text{coins of } \mathcal{A}} \left[ \text{ERR}_{k\text{-IndSelect}}(\mathbf{y}_\mathbf{u}, \mathcal{A}(\mathbf{y}_\mathbf{u})) \leq \gamma \right] \geq \frac{2}{3} \) for all \( \mathbf{u} \in [d]^k \). This means that for all \( \mathbf{u} \in [d]^k \),

\[ \Pr_{\text{coins of } \mathcal{A}} \left[ \mathcal{A}(\mathbf{y}_\mathbf{u}) = \mathbf{u} \right] \geq \frac{2}{3}. \]
For all \( u, u' \in [d] \), by group privacy, \( A(y_u) \approx_{(\gamma \varepsilon_n, 0)} A(y_{u'}) \), which implies that
\[
\Pr[A(y_u) = u'] \geq e^{-\gamma \varepsilon_n} \Pr[A(y_{u'}) = u'] \geq \frac{2}{3} e^{-\gamma \varepsilon_n}.
\]
\[
(5)
\]
Since the probability of any event is at most 1,
\[
1 \geq \Pr_{\text{coins of } A}[A(y_u) \neq u] = \sum_{u' \neq u} \Pr[A(y_u) = u'] \geq \frac{2}{3} e^{-\gamma \varepsilon_n} (d^k - 1),
\]
where the last inequality holds by (5). We get that \( e^{\gamma \varepsilon_n} \geq \frac{d^k - 1}{2} \cdot \frac{3}{2} \), and thus \( n = \Omega \left( \frac{k \log d}{\varepsilon} \right) \).

**D.1. Proof of Theorem 4.1**

Let \( \mathcal{M} \) be an \((\varepsilon, \delta)\)-DP and \((\alpha, T)\)-accurate mechanism for SumSelect in the continual release model with nonadaptively chosen inputs. We use \( \mathcal{M} \) to construct an \((\varepsilon, \delta)\)-DP algorithm \( A \) that is \((\frac{\varepsilon}{3}, n)\)-accurate for \( k\text{-IndSelect}_{d} \) 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 \( \text{SumSelect}_{d} \) in the continual release model with nonadaptively chosen inputs and \( y \) be a dataset with \( n \) records from \([0, 1]^d \). A naive approach to solving \( k\text{-IndSelect}_{d} \) 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 \( \text{SumSelect}_{d} \) and run a single instantiation of \( \mathcal{M} \) for about \( nk \) time steps, where each datapoint in \( y \) is sent to \( \mathcal{M} \) only once. This approach doesn’t suffer from privacy degradation.

Algorithm \( A \) proceeds in \( k \) stages; the \( r \)-th stage is dedicated to selecting the coordinate with the maximum sum in the \( r \)-th block. In the first stage, \( A \) streams \( y \) to \( \mathcal{M} \). In order to select the coordinate with the maximum sum from the first block, \( A \) then sends \( 2n \) records of the form \((1^d 0^d \ldots 0^d)\) to \( \mathcal{M} \). Then the sum of the coordinates in the first block of \( y \) 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, \( A \) sends \( 2n \) records of the form \((0^d 1^d \ldots d^d)\) 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, \( A \) sends \( 2n \) records of the form \((0^d 1^d 0^d \ldots 0^d)\) to \( \mathcal{M} \). At the end of the second stage, \( \mathcal{M} \) selects the coordinate with the maximum sum in the second block. Algorithm \( A \) proceeds similarly for every block.

The details of the algorithm appear in Algorithm 3. For ease of indexing, \( A \) sends all-zero records in time steps \( n + 1 \) to \( 2n \) in Step 4 of Algorithm 3, to ensure that all stages have \( 4n \) time steps.

**Algorithm 3 Batch algorithm \( A \) for \( k\text{-IndSelect} \)**

1. **Input**: \( k, y = (y_1, \ldots, y_n) \in X^n \), where \( X = \{0, 1\}^{dk} \), and black-box access to mechanism \( \mathcal{M} \).
2. **Output**: \( b = (b_1, \ldots, b_k) \in [d]^k \).
3. Let \( v_j \) be a vector of length \( dk \) with \( d \) ones in coordinates \( [dj] \setminus [d(j-1)] \) and 0 everywhere else; let \( \nabla_j \leftarrow 1^{dk} - v_j \).
4. Construct a stream \( x \leftarrow y \circ (0^{dk})^n \circ (v_1)^{2n} \circ (\nabla_1)^{2n} \circ \cdots \circ (v_{k-1})^{2n} \circ (\nabla_{k-1})^{2n} \circ (v_k)^{2n} \) with \( 4kn \) records.
5. for \( t \in [T] \) do
6. Send the record \( x_t \) to \( \mathcal{M} \) and get the corresponding output \( a_t \).
7. end for
8. for \( r \in [k] \) do
9. \( b_r \leftarrow a_{4rn} - r(d - r - 1) \). If \( b_r \notin [d] \), then \( b_r \leftarrow 1 \).
10. end for
11. Output \( b \leftarrow (b_1, \ldots, b_k) \).

**Lemma D.2.** Let \( A \) be Algorithm 3. 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 \((\varepsilon, \delta)-\)differentially private and \((\alpha, T)\)-accurate for \( \text{SumSelect}_{dk} \) in the continual release model with nonadaptively chosen inputs, then batch algorithm \( A \) is \((\varepsilon, \delta)\)-differentially private and \((\frac{\alpha}{3}, n)\)-accurate for \( k\text{-IndSelect}_{d} \).

**Proof.** We start by reasoning about privacy. Fix neighboring datasets \( y \) and \( y' \) that are inputs to algorithm \( A \). Let \( x \) and \( x' \) be the streams constructed in Step 4 of \( A \) when it is run on \( y \) and \( y' \), respectively. By construction, \( x \) and \( x' \) are neighboring streams. Since \( \mathcal{M} \) is \((\varepsilon, \delta)\)-DP, and \( A \) only post-processes the outputs received from \( \mathcal{M} \), Lemma A.4 implies that \( A \) is \((\varepsilon, \delta)\)-DP.

Next, we reason about accuracy. Fix a dataset \( y \) and the corresponding data stream \( x \) sent to \( \mathcal{M} \). Consider a setting \( \tau \) of the random coins of \( A \). Since the transformation from \( \mathcal{M} \) to \( A \) is deterministic, they correspond to coins used by \( \mathcal{M} \) when \( A \) runs it as a subroutine. Let \( \alpha_{\tau} \) be the realized error of \( \mathcal{M} \) with coins \( \tau \), that is,
\[
\alpha_{\tau} = \max_{t \in [4kn]} \left( \text{ERR}_{\text{SumSelect}_{dk}}(x_{t|\tau}, a_t) \right),
\]
where \( a_t \) are the answers with coins \( \tau \). Similarly, let \( \gamma_{\tau} \) be the realized error of \( A \) with coins \( \tau \), that is,
\[
\gamma_{\tau} = \text{ERR}_{k\text{-IndSelect}_{dk}}(y, b) = \frac{1}{n} \cdot \max_{r \in [k]} \left( \text{ERR}_{\text{SumSelect}_{d}}(y((r-1)d + 1 : rd], b_r)) \right),
\]
where \( b = (b_1, \ldots, b_k) \) is the output of \( A \) run with coins \( \tau \).

The main observation in the accuracy analysis is that if \( \alpha \) is small, so is \( \gamma \). Note that if \( \alpha \geq n \), the accuracy guarantee for \( A \) is vacuous. Now assume \( \alpha < n \). For all blocks \( r \in [k] \), the sums in \( x_{(k)} \) of \( \{v_i\}_1^n \) and \( \{v_i\}_1^n \) are \( \gamma \) where \( \gamma \) is the sum over all coordinates in block \( r \) by at least 0. Consider coins \( \tau \) with \( \alpha \tau \leq \alpha \). Since \( \alpha \tau \leq \alpha \), the index \( \alpha \) is returned by \( M \) in block \( r \) for all \( r \in [k] \). Moreover, the error for each block is at most \( \frac{\alpha}{n} \). Therefore, \( \gamma \leq \frac{\alpha}{n} \leq \gamma \). Considering the probability of this event over all coins \( \tau \), we get

\[
\Pr_{\alpha \tau \subseteq A} \left[ \gamma \leq \frac{\alpha}{n} \right] \geq \Pr_{\gamma \tau \subseteq A} \left[ \gamma \leq \alpha \right] \geq \frac{2}{3},
\]

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

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 \( (e_j)_{j=1}^l \) be a record of length \( d \) with 1 in coordinate \( j \) and 0 everywhere else. Let \( x = (e_j)_{j=1}^l \) and \( x' = (e_{j'})_{j'=1}^l \). 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 \( \varepsilon \), we reduce from \( k \)-IndSelect, relying on the lower bounds for \( k \)-IndSelect from Lemma D.1. Fix \( T, d, \varepsilon \). Given an integer \( k \), the reduction of Lemma D.2 maps a batch instance of \( k \)-IndSelect, \( d \) of size \( n \) to an instance of \( e \)-Select by \( d' = dk \) and \( T = 4nk \). The reduction applies as long as \( d' = \frac{d}{T} \geq 2 \) and \( n = \frac{Td}{2} \geq 1 \) are integers. We will ignore the integrality requirement (which can be addressed by appropriate padding) and allow any \( k \) between 1 and \( \min \left( \frac{d}{T}, \frac{T}{d} \right) \).

When \( \delta = o\left( \frac{1}{T^2} \right) \), we get \( \delta = o\left( \frac{1}{n^2} \right) \) (since \( \varepsilon < 1 \) and \( T > n \)), since our reduction in Lemma D.2 preserves \( \delta \), we are in the range of \( \delta \) where Lemma D.1 Part 1 applies. This gives us a lower bound on the error of \( \min \left( \frac{T}{\sqrt{\log d}}, \frac{1}{\varepsilon} \right) \) when \( k' \) and \( d' \) are sufficiently large constants. In our setting, this translates to a lower bound of \( \Omega(\alpha_k) \) for \( \alpha_k = \min \left( \frac{T \log (2 + d'/k')}{\varepsilon}, \frac{1}{\varepsilon} \right) \). (We add 2 inside the logarithms to avoid 0 or subconstant log terms; this does not change the asymptotics.) For simplicity, we omit the dependency on \( \log(1/\delta) \) in the lower bounds.

Our goal is to select the value of \( k \in [1, \min(\frac{d}{T}, \frac{T}{d})] \) that maximizes \( \alpha_k \). For fixed \( T, d, \varepsilon, \) let \( k = k^*(T, d, \varepsilon) = \max(1, k') \) where \( k' \) denotes the largest value of \( k \) where the two terms defining \( \alpha_k \) equalize (that is, \( k \) satisfies \( k' \sqrt{k'} \log(2 + d' / k') = cT \)). We use two basic facts about \( \alpha_k \): first, for \( d \geq 1 \), the function \( \alpha_k \) is increasing on \([1, k^*] \) and decreasing on \((k^*, \infty) \).

Second, its maximum value \( \alpha_k^* \) is \( \Omega \left( \frac{T^{1/3} \log \sqrt{2^3 d}}{\varepsilon^{1/3}} \right) \). To see why this is, note that \( k^* = \left( \frac{cT}{\log (2 + d')} \right)^{2/3} \), and so \( \alpha_k^* = \frac{T^{1/3} \log (2 + d')}{\varepsilon^{1/3}} \). Consider two cases: if \( k' \leq \sqrt{d} \), then \( d' \geq \sqrt{d} \) and so \( \log(2 + d') = \Theta(\log d) \). On the other hand, if \( k' \geq \sqrt{d} \), then \( \alpha_k^* \), which is always at least \( k^* \), is bounded below by \( d^{1/4} \). Therefore the factor of \( \log(d) \) is polynomial in \( \log(\alpha_k^*) \) and absorbed by the \( \Omega \) notation.

We consider four regimes for the triple \((T, d, \varepsilon) \). Let \( C \) denote a constant such that Lemma D.1 applies for \((d, \varepsilon) \geq C \).

(a) \( k^*(T, d, \varepsilon) = 1 \): In this case, \( \alpha_k \) is maximized at \( k = 1 \). Since setting \( k' = 1 \) is sufficiently large constant does not change the asymptotics of the lower bound, we can set \( k \) sufficiently large for Lemma D.1 to apply, and obtain a lower bound of \( \Omega(T/k) = \Omega(T) \).

(b) \( k^*(T, d, \varepsilon) > \min(\frac{d}{T}, \frac{1}{\varepsilon}) \) and \( C < d \): In this case, we set \( k' = d/C \). We get a lower bound of \( \alpha_k = \frac{T \log(2 + d/k)}{\varepsilon} \) (since \( k' \leq k^* \)), which is \( \Omega(\frac{2T}{\varepsilon}) \).

(c) \( k^*(T, d, \varepsilon) > \min(\frac{d}{T}, \frac{1}{\varepsilon}) \) and \( C > d \): 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 \( \alpha_k^* = \frac{T^{1/3} \log(2^3 d)}{\varepsilon^{1/3}} \), we get that \( \varepsilon > 1 \) (for sufficiently large \( T \)), contradicting our assumptions.

(d) \( k^*(T, d, \varepsilon) \in [1, \min(\frac{d}{T}, \frac{1}{\varepsilon})] \): In this case, we set \( k = k^* \) and obtain a lower bound of \( \alpha_k^* = \frac{T^{1/3} \log \frac{2^3 d}{\varepsilon}}{\varepsilon^{1/3}} \). (Note that if \( k^* \) is too small, we can set \( k = ck^* \) for a sufficiently large constant \( c \) without changing the asymptotics, as in part a). Thus, for all possible relationships between \( T, d \) and \( \varepsilon \), 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 \( k \in [1, \min(\frac{d}{T}, \frac{T}{d})] \), we obtain a lower bound of \( \Omega(\alpha_k) \) for \( \alpha_k = \min \left( \frac{T \log (2 + d')}{\varepsilon}, \frac{1}{\varepsilon} \right) \). The remaining calculations parallel the case where \( \delta > 0 \), except that now
\[
\alpha_k^* = \Omega\left( \frac{T \log d}{\varepsilon} \right). \]

\( \Box \)

**E. D. Omitted from Section 5**

**E.1. Formal Statements**

In this subsection, we state theorems that summarize the performance guarantees of our mechanisms for MaxSum and SumSelect. We prove these theorems in the following subsections. The theorems provide a stronger privacy guarantees...
guarantee than \((\varepsilon, \delta)\)-DP, specifically, concentrated differential privacy. We state implications for \((\varepsilon, \delta)\)-DP in Corollary E.3. The upper bounds in our theorems are attained by two simple mechanisms: one uses the binary tree mechanism and the other recomputes the target function at regular intervals. Prior to our work, it was not known that these mechanisms are private in the setting with adaptively chosen inputs. As mentioned in Section 1.1, a proof of these mechanisms are private in the setting with adaptively chosen inputs.

**Theorem E.1** (zCDP, Binary-Tree-Based Mechanisms). For all \(\rho \in \{0, 1\}, d \in \mathbb{N}, \) and sufficiently large \(T > 0,\) there exist \(\rho\)-zCDP mechanisms \(M\) and \(M'\) in the continual release model with adaptively chosen inputs such that \(M\) is \((\alpha, T)\)-accurate for \(\text{MaxSum}_{d}\) and \(M'\) is \((\alpha, T)\)-accurate for \(\text{SumSelect}_d\), where

\[
\alpha = O\left(\frac{\sqrt{d \log T}}{\sqrt{\delta}}\right).
\]

The next theorem uses the idea of recomputing at regular intervals, which applies quite generally. Item 1 of Theorem E.2 applies for general sensitivity-1 functions (which include \(\text{MaxSum}_d\)); a similar result holds for bounded-sensitivity functions with output space \(\mathbb{R}^d\).

**Theorem E.2** (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 \(\ell_2\)-sensitivity at most 1, there exist \(\rho\)-zCDP mechanisms \(M\) and \(M'\) in the continual release model with adaptively chosen inputs such that

1. Mechanism \(M\) is \((\alpha, T)\)-accurate for \(f\) for \(\alpha = O\left(\min\left\{\frac{T \log T}{\rho}, T\right\}\right)\);
2. Mechanism \(M'\) is \((\alpha, T)\)-accurate for \(\text{SumSelect}_d\) for \(\alpha = O\left(\min\left\{\frac{T^{1/3} \log^{2/3}(dT)}{\rho^{1/3}}, T\right\}\right)\).

We combine Theorems E.1–E.2, use the conversion from zCDP to \((\varepsilon, \delta)\)-DP from Lemma A.15, and substitute \(\rho = \frac{e^2}{16 \log(1/\delta)}\) to get the following corollary.

**Corollary E.3.** For all \(\varepsilon \in (0, 1], \delta \in (0, \frac{1}{2}], d \in \mathbb{N}, \) and sufficiently large \(T > 0,\) there exist \((\varepsilon, \delta)\)-DP mechanisms \(M\) and \(M'\) in the continual release model with adaptively chosen inputs such that

1. \(M\) is \((\alpha, T)\)-accurate for \(\text{MaxSum}_{d}\) for \(\alpha = O\left(\min\left\{\sqrt{T \log (1/\delta) \log T} \frac{1}{\varepsilon^{1/3}}, \frac{d \log(dT) \log(1/\delta) \log T}{\varepsilon}\right\}\right)\);
2. \(M'\) is \((\alpha, T)\)-accurate for \(\text{SumSelect}_d\) for \(\alpha = O\left(\min\left\{\sqrt{T \log^2(dT) \log(1/\delta) \log T} \frac{1}{\varepsilon^{2/3}}, \frac{d \log(dT) \log(1/\delta) \log T}{\varepsilon}\right\}\right)\).

### E.2 Algorithms based on the Binary Tree Mechanism

In this section, we prove Theorem E.1 for \(\text{SumSelect}_d.\) Theorem E.1 for \(\text{MaxSum}_d\) 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 \(\text{SumSelect}_d\) on a dataset with \(d\) attributes, we use the binary tree mechanism from Chan et al. (2011); Dwork et al. (2010a) to privately sum each of the attributes of the records \(x_{[i]}\) 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 \(\{i, \ldots, j\}.\)
Algorithm 4 Mechanism $M$ for SumSelect in continual release model with adaptively chosen inputs

1: Input: time horizon $T \in \mathbb{N}$, privacy parameter $\rho$, stream $x = (x_1, \ldots, x_T) \in \mathcal{X}^T$, where $\mathcal{X} = \{0,1\}^d$.
2: Output: stream $(a_1, \ldots, a_T) \in [d]^T$.
3: Init: Construct a complete binary tree with $T$ leaves labeled $v_{[1:t]}, \ldots, v_{[T:T]}$. Label every internal node $v_{[\ell:r]}$ if the subtree rooted at that node has leaves $v_{[\ell]}$, $\ldots$, $v_{[r]}$. Initialize the partial sum $s_{[\ell:r]} \leftarrow 0$ for each node $v_{[\ell:r]}$ in the tree.
4: for $t = 1$ to $T$ do
5: Get record $x_t$ from $Adv$.
6: $\triangleright$ Compute noisy sums for nodes completed at time $t$
7: for each node $v_{[\ell:t]}$ do
8: Draw noise $Z \sim \mathcal{N}(0, \sigma^2 2^{d \times d})$, where $\sigma = \sqrt{\frac{d(\log T + 1)}{2^p}}$, and set $s_{[\ell:t]} \leftarrow s_{[\ell:t]} + \sigma Z$.
9: $\triangleright$ Output Steps:
10: $I_t \leftarrow$ collection of at most $\log t + 1$ intervals that partition $[1 : t]$, where each interval labels a node in the binary tree. (See Remark E.2.)
11: $s_{[\ell:r]} \leftarrow \sum_{t \in I_t} s_{[\ell:t]}$.
12: Output $a_t \leftarrow \operatorname{arg max}_{j \in [d]} s_{[\ell]}$.

At the high level, the binary tree mechanism constructs a complete binary tree with $T$ leaves. The leaves correspond to the input records $x_{[t]}$, where each record $x_t \in \{0,1\}^d$. 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 $\mathcal{N}(0, \sigma^2 2^{d \times d})$ with $\sigma = \sqrt{\frac{d(\log T + 1)}{2^p}}$. The algorithm that releases the noisy sum is $\rho$-zCDP. Since each $x_t$ participates in only $\log_2 T + 1$ sums in the tree, by adaptive composition of zCDP (Lemma A.13), the complete mechanism is $\rho$-zCDP (Theorem E.1). 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 $(\alpha,T)$-accurate for $\alpha \approx O\left(\frac{\sqrt{d} \log T \log(Td)}{\sqrt{p}}\right)$. The formal description of the algorithm appears in Algorithm 4. The algorithm uses a dyadic decomposition (described in Remark E.2) 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 E.7 and E.8, respectively, which together prove Theorem E.1 for SumSelect.

Lemma E.7. For all $\rho \in \mathbb{R}^+, d, T \in \mathbb{N}$, mechanism $M$ described in Algorithm 4 is $\rho$-zCDP in the continual release model with adaptively chosen inputs.

Proof. Consider an adversary $Adv$ interacting with the privacy game $\Pi_{M_{\rho},Adv}$. We want to argue that the adversary’s view is $\rho$-close in the two versions of the privacy game (for the two possible values of side $\in \{L, R\}$). We will achieve this by introducing a $\rho$-zCDP mechanism $M_{gauss}$ with input side and reducing our goal to the privacy of $M_{gauss}$. For this, we use a simulation argument similar to those used in cryptography. Specifically, our proof defines two algorithms: (a) a $\rho$-zCDP mechanism $M_{gauss}$ that gets input side $\in \{L, R\}$ and (b) a simulator $Sim$ with query access to $M_{gauss}$ that does not know the value of side. The simulator $Sim$ interacts with adversary $Adv$ and satisfies a key guarantee:

The view of the adversary $Adv$ in its interaction with $Sim$ is identically distributed to its view in the privacy game $\Pi_{M_{\rho},Adv}$, defined in Algorithm 1. (Figure 1 illustrates the structure of these two kinds of interaction.)

Since the simulator’s outputs to $Adv$ are a post-processing of the query responses from $M_{gauss}$, we can argue that the adversary’s view is $\rho$-close in the two versions of the privacy game $\Pi_{M_{\rho},Adv}$.

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

The mechanism $M_{gauss}$ (described in Algorithm 6) gets an input side $\in \{L, R\}$. It receives at most $\log T + 1$
queries of the form \(v(L), v(R)\) from Sim to which it responds with \(p = v(\text{side}) + Z\) where the noise \(Z\) is drawn from \(\mathcal{N}(0, \sigma^2 T^d)\) for \(\sigma = \sqrt{\frac{\log T + 1}{2\rho}}\). Observe that if \(\mathcal{M}_{\text{gauss}}\) has only a single interaction with Sim and outputs a single noised value, then by the privacy guarantee of the Gaussian mechanism (Lemma A.14), \(\mathcal{M}_{\text{gauss}}\) is \(\frac{\rho}{\log T + 1}\)-zCDP. This can be seen by imagining that \(\mathcal{M}_{\text{gauss}}\) is computing a function \(f(\text{side}) = x_\text{side}^d\) and observing that the \(\ell_2\)-sensitivity of \(f\) is \(\sqrt{d}\). Since there are \(\log T + 1\) interactions between \(\mathcal{M}_{\text{gauss}}\) and Sim, \(\mathcal{M}_{\text{gauss}}\) is an adaptive composition of \(\log T + 1\) algorithms, each of which is \(\frac{\rho}{\log T + 1}\)-zCDP. By Lemma A.13 on composition, \(\mathcal{M}_{\text{gauss}}\) is \(\rho\)-zCDP.

The simulator Sim (described in Algorithm 5) interacts with the adversary without knowing the input side \(\in \{L, R\}\) that is given to \(\mathcal{M}_{\text{gauss}}\). It queries \(\mathcal{M}_{\text{gauss}}\) 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 \(\Pi_{\mathcal{M}, \text{Adv}}\) 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 Sim without any help from \(\mathcal{M}_{\text{gauss}}\). For the nodes whose interval does contain \(t^*\), the simulator sends \((x_{\text{L}}^{(L)}(t), x_{\text{R}}^{(R)}(t))\) to \(\mathcal{M}_{\text{gauss}}\) and gets a noised value of \(x_{\text{side}}^{(L)}\). 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 \(\Pi_{\mathcal{M}, \text{Adv}}\) would.

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

\[\mathcal{M}_{\text{gauss}} \text{ is } \rho\text{-zCDP} \implies V_{\mathcal{M}, \text{Adv}} \sim_{\rho} V_{\mathcal{M}_{\text{gauss}}, \text{Adv}}.\]

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 Sim (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 \(t^*\).

**Lemma E.8.** For all \(\rho > 0\) and sufficiently large \(T \in \mathbb{N}\), mechanism \(\mathcal{M}\) is \((\alpha, T)\)-accurate for SumSelect in the continual release model with adaptively chosen inputs for \(\alpha = O\left(\sqrt{\frac{3\log T \sqrt{\log(Td)}}{\sqrt{\rho}}}\right)\).

**Proof.** Consider any adversarial process \(\text{Adv}\) interacting with \(\mathcal{M}\). We first argue that, at every timestep \(t\), the random variable \(\text{ERR}_{\text{SumSelect}}(x_{[t]}, a_t)\) 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, \(\max_{t \in [T]} \text{ERR}_{\text{SumSelect}}(x_{[t]}, a_t)\) is not too large.

First, at any timestep \(t\), let \(\text{sum}_t\) represent the vector of noisy sums defined in Step 10 of Algorithm 4. Therefore,
Algorithm 5: Simulator Sim for the proof of Lemma E.7

1: **Input:** time horizon $T \in \mathbb{N}$, privacy parameter $\rho \in \mathbb{R}^+$, black-box access to an adversary $\mathcal{A}_{adv}$ and a mechanism $\mathcal{M}_{gauss}$.
2: **Output:** stream $(a_1, \ldots, a_T) \in [d]^T$.

3: **$\mathcal{A}_{adv}$:** At each timestep $t \in [T] \setminus \{t^*\}$, $\mathcal{A}_{adv}$ provides $\sim$ with record $x_t \in X$, where $X = \{0, 1\}^d$.
   - At the challenge timestep $t^*$ (chosen by $\mathcal{A}_{adv}$), it provides records $x_t^{(L)}, x_t^{(R)} \in X$.
   - At each timestep $t \in [T]$, Sim provides $\mathcal{A}_{adv}$ with output $a_t \in [d]$.
4: $\mathcal{M}_{gauss}$: Sim exchanges $\log_2 T + 1$ messages with $\mathcal{M}_{gauss}$.
5: **Init:** Perform Step 1 (the initialization phase) of Algorithm 4.
6: $j \leftarrow 1$.
7: for $t \in [T]$ do
   8: if $t = t^*$ then
      9: Get input $(x_t^{(L)}, x_t^{(R)})$ from $\mathcal{A}_{adv}$.
     10: for $i \in [\log_2 T + 1]$ do
        11: Send $(x_t^{(L)}, x_t^{(R)})$ to $\mathcal{M}_{gauss}$, and get back a response $p_i$.
    12: end for
   13: else
   14: Get record $x_t$ from $\mathcal{A}_{adv}$.
   15: end if
    16: for each node $v_{i,t}$ do
       17: if $t^* \notin \{t : t\}$ where $\{t : t\}$ denotes the integers $\{\ell : t\}$ then
          18: Draw noise $Z \sim \mathcal{N}(0, \sigma^2 2^{t \times d})$, where $\sigma = \sqrt{\frac{d(\log T + 1)}{2\rho}}$.
         19: $s_{i,t} \leftarrow Z + \sum_{i=\ell}^{t} x_i$
    20: else
    21: $v_{i,t} \leftarrow \sum_{i \in \{\ell : t\} \setminus \{t^*\}} x_i + p_j$
    22: $j \leftarrow j + 1$.
    23: end if
    24: end if
26: end for

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

1: **Input:** side $\in \{L, R\}$ (not known to $\mathcal{Sim}$).
2: **Output:** A natural number.
3: for $i = 1$ to $\log T + 1$ do
4: Get records $v_i^{(L)}, v_i^{(R)} \in \{0, 1\}^d$ from $\mathcal{Sim}$.
5: Draw noise from a multivariate Gaussian distribution $Z \sim \mathcal{N}(0, \sigma^2 2^{t \times d})$, where $\sigma = \sqrt{\frac{d(\log T + 1)}{2\rho}}$.
6: Output $v_i^{(side)} \leftarrow Z$
7: end for
mechanism used is the same as Algorithm 4, except that in Step 7, $Z$ is drawn from $\text{Lap}(\frac{d\log(T+1)}{\varepsilon})$ instead of a Gaussian distribution. The privacy proof is exactly as in Lemma E.7, 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 of a composition theorem for $\rho$-$z$CDP. The accuracy proof closely follows that of Lemma E.8, 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 $\text{Lap}\left(\frac{d\log(T+1)}{\varepsilon}\right)$. 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 F.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 $\frac{1}{T^m}$. 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 $\frac{1}{T^m}$, 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.

Theorems E.1 and E.4 for MaxSum$_d$ are proved analogously. The main difference is that we output $\max_{j \in [d]} \text{sum}_t[j]$ instead of $\arg\max_{j \in [d]} \text{sum}_t[j]$ in Step 11 of Algorithm 4.

### E.3. Algorithms that Recompute at Regular Intervals

In this section, we prove Item 1 of Theorem E.2 for sensitivity-1 functions. The proof of Item 2 of Theorem E.2 builds on the same idea of recomputing SumSelect$_d$ every $T/m$ timesteps, but it uses the report noisy max (with exponential noise) algorithm for SumSelect$_d$ (McKenna & Sheldon, 2020) 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 continual release model with adaptively chosen inputs

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

#### Claim E.9. For all $\rho, T > 0$, $r \in [T−1]$, mechanism $\mathcal{M}$ defined in Algorithm 7 is $\rho$-$z$CDP in the continual release model with adaptively chosen inputs.

**Proof.** Consider an adversary $\mathcal{Adv}$ interacting with $\mathcal{M}$. We define a mechanism $\mathcal{M}_{\text{comp}}$, similar to Algorithm 6, and a simulator $\text{Sim}$ that interacts with the adversary $\mathcal{Adv}$ such that the view of adversary $\mathcal{Adv}$ in the interaction with $\mathcal{M}_{\text{comp}}$ and $\text{Sim}$ is identically distributed to its view in the privacy game $\Pi_{\mathcal{M}_{\text{comp}}, \mathcal{Adv}}$, defined in Algorithm 1.

#### Algorithm 8 Mechanism $\mathcal{M}_{\text{comp}}$

1. **Input:** side $\in \{L, R\}$ (not known to $\text{Sim}$)
2. **Output:** A natural number.
3. Get neighboring datasets $y^{(L)}, y^{(R)} \in \{0,1\}^d$ and a function $f$ with $\ell_2$ sensitivity at most 1 from $\text{Sim}$.
4. Draw noise $Z \sim \mathcal{N}(0, \sigma^2)$, where $\sigma = \sqrt{\frac{m}{2p}}$.
5. Output $f(y^{(\text{side})}) + Z$

The mechanism $\mathcal{M}_{\text{comp}}$ is defined in Algorithm 8. Since the function $f$ has $\ell_2$ sensitivity at most 1, then by the privacy of the Gaussian mechanism, and since the variance of the noise added is $\frac{m}{2p}$. $\mathcal{M}_{\text{comp}}$ is $\frac{\rho}{m}$-$z$CDP with respect to the dataset consisting of side $\in \{L, R\}$.

The simulator $\text{Sim}$ (described in Algorithm 9) gets inputs from $\mathcal{Adv}$, but it does not know the input side $\in \{L, R\}$ that is given to $\mathcal{M}_{\text{comp}}$. It interacts with $\mathcal{M}_{\text{comp}}$ to provide outputs to the adversary $\mathcal{Adv}$. The aim of the Simulator is to mimic the behaviour of $\Pi_{\mathcal{M}_{\text{comp}}, \mathcal{Adv}}$ 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,
Algorithm 9 Simulator Sim for the proof of Claim E.9

1: Input: time horizon \( T \), privacy parameter \( \rho > 0 \), recompute period \( r \in [T - 1] \), function \( f \). Sim also has black-box access to an adversary \( Adv \) and a process \( M_{\text{comp}} \).

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

3: \( Adv \): At each timestep \( t \in [T] \setminus \{t^*\} \), \( Adv \) provides Sim with record \( x_t \in \mathcal{X} \). At the challenge timestep \( t^* \) (chosen by \( Adv \)), it provides two records \( x_t^{(L)}, x_t^{(R)} \in \mathcal{X} \). At every timestep \( t \leq T \), Sim provides \( Adv \) with output \( a_t \in \mathbb{R} \).

4: \( M_{\text{comp}} \): Sim exchanges \( T/r \) messages with \( M_{\text{comp}} \).

5: Initialization: \( m \leftarrow \lceil \frac{T}{r} \rceil \), \( j \leftarrow 1 \).

6: For timesteps \( t < t^* \), run mechanism \( M \) in Algorithm 7 with inputs from \( Adv \), with the same \( T, \rho, r, f \).

7: Let \( a_t \) be \( M \)'s output at timestep \( t \).

8: for \( t \geq t^* \) do

9: if \( t = t^* \) then

10: Get input \( (x_t^{(L)}, x_t^{(R)}) \) from \( Adv \).

11: else

12: Get record \( x_t \) from \( Adv \).

13: end if

14: if \( t \mod r = 1 \) then

15: For each side \( \in \{L, R\} \), let \( y_t^{(\text{side})} = \{x_1, \ldots, x_{t-1}, x_t^{(\text{side})}, x_{t+1}, \ldots, x_t\} \);

16: \( a_t \leftarrow M_{\text{comp}} \left( f, y_t^{(L)}, y_t^{(R)} \right) \).

17: end else

18: \( q \leftarrow \lfloor \frac{1}{r} \rfloor \); output \( a_t \leftarrow a_{q+1} \).

19: end if

20: end for

Sim sends \( M_{\text{comp}} \) the function \( f \) as well as neighboring datasets \( y_t^{(L)}, y_t^{(R)} \) defined by

\[
y_t^{(\text{side})} = \{x_1, \ldots, x_{t-1}, x_t^{(\text{side})}, x_{t+1}, \ldots, x_t\}.
\]

Since Sim queries \( M_{\text{comp}} \) at most \( m \) times, by adaptive composition, the output transcript of \( M_{\text{comp}} \) is \( \rho \)-zCDP with respect to the dataset consisting of \( \mathcal{X} \).

The view of the adversary \( Adv \) in the real privacy game \( \Pi_{M, Adv} \) is identically distributed to its view in the interaction with \( M_{\text{comp}} \) and Sim. Furthermore, the view of the adversary \( Adv \) when interacting with Sim and \( M_{\text{comp}} \) is simply a post-processing of the outputs provided to it by Sim, which are a post-processing of the outputs provided to Sim by \( M_{\text{comp}} \). As argued previously, the output of \( M_{\text{comp}} \) when side = \( L \) is \( \rho \)-close to its output transcript when side = \( R \). Hence we have that \( V_{M, Adv}^{(L), (R)} \succeq_{\rho} V_{M_{\text{comp}}}^{(L), (R)} \).

Claim E.10. Fix \( \rho > 0 \), sufficiently large \( T > 0 \), and \( 2 \leq m \leq T \). Let \( f : \mathcal{X}^* \rightarrow \mathbb{E} \) be a function with \( \ell_2 \)-sensitivity at most 1. Then mechanism \( M \), defined in Algorithm 7 is \((\alpha, T)\)-accurate for \( f \) in the continual release model with adaptively chosen inputs where \( \alpha = \frac{T}{m} + \sqrt{\frac{10m \log m}{\rho}} \).

Proof. Consider any adversarial process \( Adv \) interacting with \( M \). Fix a timestep \( t \in [T] \). Consider time horizon \( T \) divided into \( m \) stages, where the stage \( k \) is from timestep \((k - 1)r + 1 \) to \( kr \). Let timestep \( t \) be in stage \( k \). Intuitively, since \( M \), 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,

\[
\text{ERR}_f(x_{[t]}, a_t) = |a_t - f(x_{[t]})| \\
\leq |f(x_{[t]}) - f(x_{[(k-1)r+1]})| + |a_t - f(x_{[(k-1)r+1]})| \\
\leq T/m + |a_{(k-1)r+1} - f(x_{[(k-1)r+1]})| \leq \frac{T}{m} + |Z_k|.
\]

The second inequality above holds because the \( \ell_2 \)-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 7 and 10 in Algorithm 7. Finally, observe that \( Z_k \) for \( k \in [m] \) are mutually independent Gaussian random variables with mean 0 and standard deviation \( \sqrt{\frac{m}{2r}} \). Hence, applying Lemma F.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 \( m \geq 2 \),

\[
\Pr_{A, Adv} \left( \max_{t \in [T]} \text{ERR}_f(x_{[t]}, a_t) \geq \frac{T}{m} + \sqrt{\frac{10m \log m}{\rho}} \right) \\
= \Pr_{A, Adv} \left( \max_{k \in [m]} |Z_k| \geq \sqrt{\frac{10m \log m}{\rho}} \right) \leq \frac{2}{m^3} \leq \frac{1}{3}.
\]

Proof of Item 1 in Theorem E.2. By Claim E.9, the mechanism \( M \) is \( \rho \)-zCDP in the continual release model with adaptively chosen inputs.

For \( \rho \leq \frac{\log T}{r^2} \), 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 \( \rho > \frac{\log T}{r^2} \), by Claim E.10, mechanism \( M \) is \((\alpha, T)\)-accurate for \( f \) in the continual release model with adaptively chosen inputs, where \( \alpha = T/m + 10\sqrt{\frac{m \log m}{\rho}} \). Setting \( m = \left[ \frac{10^{1/3} \rho^{2/3}}{\log^{1/3} T} \right] \) gives \( \alpha = O \left( \min \left\{ T, \frac{10 \log T}{\rho} \right\} \right) \), where the min comes from the option of using the trivial mechanism. \( \square \)
Proof Sketch of Item 1 in Theorem E.5. The mechanism $\mathcal{M}$ used is a variant of Algorithm 7. The only difference is that in Step 6, 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 E.9, with the main difference being that instead of using a composition theorem for $\rho$-zCDP, we instead use that the composition of $m$ mechanisms that are $(\varepsilon, 0)$-DP is $(\varepsilon, 0)$-DP.

For accuracy, we can prove a claim phrased exactly as Claim E.10, with $\alpha = \frac{T}{m} + \frac{m}{\varepsilon} \log m + 2 \log T$ instead of $\alpha = \frac{T}{m} + \sqrt{\frac{10m \log m}{p}}$. The proof is similar, with the only difference being that instead of using Lemma F.2 on the maximum of i.i.d. Gaussian random variables, we instead use Lemma F.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 $m = \lfloor \frac{\sqrt{eT}}{\varepsilon} \rfloor$ in the accuracy claim gives $\alpha = O\left(\frac{\sqrt{T}}{\varepsilon} \log T\right)$. 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 $(\alpha, T)$-accurate for $f$ in the continual release model with adaptively chosen inputs with $\alpha = O(T)$. This completes the proof.

Proof Sketch of Item 2 in Theorems E.2 and E.5. We sketch the proof of Item 2 of Theorem E.2. The proof of Item 2 of Theorem E.5 is essentially the same. The upper bound mechanism $\mathcal{M}$ used for this proof is a variant of Algorithm 7 where we recompute $\text{SumSelect}_d$ using the exponential mechanism (McSherry & Talwar, 2007) with $\varepsilon' = \sqrt{2d/m}$ (for Item 2 of Theorem E.5 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 $\text{SumSelect}_d$ 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 E.9. 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 $x_{i}^{(L)}$ and $x_{i}^{(R)}$ 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 $x_{i}^{(side)}$ 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 side = $L$ and when side = $R$ follows directly from the privacy of the exponential mechanism and adaptive composition (Dwork et al., 2010c; Bun & Steinke, 2016).

For accuracy, we prove a claim akin to Claim E.10, with $\alpha = \frac{T}{m} + 2 \sqrt{\frac{m}{2}\log d + 5 \log m}$. The proof is similar to that of Claim E.10; here, we define $|Z_k|$ as the error incurred by the $k^{th}$ instantiation of the exponential mechanism, and use Lemma A.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 $\alpha = \frac{T}{m} + 2 \sqrt{\frac{m}{2}\log d + 5 \log m}$ with probability at most $\frac{1}{m^2}$.

For $\rho > (\log(dT))^2$, by the accuracy claim, mechanism $\mathcal{M}$ is $(\alpha, T)$-accurate for $f$ in the continual release model with adaptively chosen inputs, where $\alpha = \frac{T}{m} + 2 \sqrt{\frac{m}{2}\log d + 5 \log m}$. Setting $m = \lfloor \frac{(e^{1/3}-1/2)^{2/3}}{(\log(dT)^2)^{2/3}} \rfloor$ yields $\alpha = O\left(\frac{T^{2/3}\log(dT)^{2/3}}{\rho^{1/3}}\right)$. Finally, for $\rho \leq (\log(dT))^2$, consider the mechanism that doesn’t touch the data and always outputs 0. It is clearly 0-zCDP, and has $\alpha = O(T)$. $\square$

F. Useful Concentration Inequalities

Lemma F.1. For all random variables $R \sim \mathcal{N}(0, \sigma^2)$, \[ \Pr[|R| > \ell] \leq 2e^{-\ell^2 / 2\sigma^2}. \]

Lemma F.2. Consider $m$ random variables $R_1, \ldots, R_m \sim \mathcal{N}(0, \sigma^2)$. Then \[ \Pr\left[ \max_{i \in [m]} |R_i| > \ell \right] \leq 2m e^{-\ell^2 / 2\sigma^2}. \]

Proof. By a union bound and Lemma F.1,
\[ \Pr\left[ \max_{i \in [m]} |R_i| > \ell \right] = \Pr(\exists i \in [m] \text{ such that } |R_i| > \ell) \leq \sum_{i=1}^{m} \Pr(|R_i| > \ell) \leq \sum_{i=1}^{m} 2e^{-\ell^2 / 2\sigma^2} = 2me^{-\ell^2 / 2\sigma^2}. \]

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 F.3. Fix $m \in \mathbb{N}$, $\lambda > 0$. Consider $m$ random variables $R_1, \ldots, R_m \sim \text{Lap}(\lambda)$. Then for all $a > 0$, \[ \Pr\left( \max_{i \in [m]} |R_i| > \lambda(\log m + \log a) \right) \leq e^{-a}. \]