Lower Bounds for Differential Privacy Under Continual Observation and Online Threshold Queries

Edith Cohen* Xin Lyu† Jelani Nelson‡ Tamás Sarlós§ Uri Stemmer¶

February 29, 2024

Abstract

One of the most basic problems for studying the "price of privacy over time" is the so called *private counter problem*, introduced by Dwork et al. (2010) and Chan et al. (2010). In this problem, we aim to track the number of *events* that occur over time, while hiding the existence of every single event. More specifically, in every time step $t \in [T]$ we learn (in an online fashion) that $\Delta_t \geq 0$ new events have occurred, and must respond with an estimate $n_t \approx \sum_{j=1}^{t} \Delta_j$. The privacy requirement is that *all of the outputs together*, across all time steps, satisfy *event level* differential privacy.

The main question here is how our error needs to depend on the total number of time steps T and the total number of events n. Dwork et al. (2015) showed an upper bound of $O\left(\log(T) + \log^2(n)\right)$, and Henzinger et al. (2023) showed a lower bound of $\Omega\left(\min\{\log n, \log T\}\right)$. We show a new lower bound of $\Omega\left(\min\{n, \log T\}\right)$, which is tight w.r.t. the dependence on T, and is tight in the sparse case where $\log^2 n = O(\log T)$. Our lower bound has the following implications:

- We show that our lower bound extends to the *online thresholds* problem, where the goal is to privately answer many "quantile queries" when these queries are presented one-by-one. This resolves an open question of Bun et al. (2017).
- Our lower bound implies, for the first time, a separation between the number of mistakes obtainable by a private online learner and a non-private online learner. This partially resolves a COLT'22 open question published by Sanyal and Ramponi.
- Our lower bound also yields the first separation between the standard model of private online learning and a recently proposed relaxed variant of it, called *private online prediction*.

1 Introduction

Differential privacy (DP), introduced by Dwork et al. (2006), is a mathematical definition for privacy that aims to enable statistical analysis of datasets while providing strong guarantees that individual-level information does not leak. Most of the academic research on differential privacy

*Google Research and Tel Aviv University. edith@cohenwang.com
†UC Berkeley and Google Research. lyuxin1999@gmail.com
‡UC Berkeley and Google Research. minilek@alum.mit.edu
§Google Research. stamas@google.com
¶Tel Aviv University and Google Research. u@uri.co.il

has focused on the "offline" setting, where we analyze a dataset, release the outcome (in a privacy preserving manner), and conclude the process. In contrast, most of the industrial applications of differential privacy are "continual" and involve re-collecting and re-aggregating users' data, say on a daily basis. In this work we study one of the most popular models for arguing about this, called the *continual observation model* of differential privacy (Dwork et al., 2010b; Chan et al., 2010). Algorithms in this model receive a stream of sensitive inputs and update their outcome as the stream progresses while guaranteeing differential privacy. This model abstracts situations where the published statistics need to be continually updated while sensitive information is being gathered.

Before describing our new results, we define our setting more precisely. Consider an algorithm $\mathcal{A}$ that on every time step $t \in [T]$ obtains a (potentially empty) dataset $D_t \in X^*$ and outputs an answer y_t. We say that $\mathcal{A}$ solves a problem $P : X^* \to \mathbb{R}$ if for every sequence of datasets $D = (D_1, \ldots, D_T)$, with high probability over $\mathcal{A}$'s coins, for every $t \in [T]$ it holds that $y_t \approx P(D_1 \circ \cdots \circ D_t) \triangleq p_t$. We measure quality by the ℓ_∞ norm, that is, for a sequence of outputs $y = (y_1, \ldots, y_T)$ we denote $\text{error}_P(D, y) = \max_{t \in [T]} |y_t - p_t|$. The privacy requirement is that the *sequence of all T outputs* should be differentially private w.r.t. D. That is, the outcome distribution of the y should be insensitive to a change limited to one elements in one of the datasets D_t. Formally,

Definition 1.1 (Dwork et al. (2006))**.** *Given a sequence of datasets $D = (D_1, \ldots, D_T)$ and an algorithm $\mathcal{A}$, we write $\texttt{Releases}_{\mathcal{A}}(D)$ to denote the sequence of T outcomes returned by $\mathcal{A}$ during the execution on D. Algorithm $\mathcal{A}$ is* (ε, δ)-differentially private *if for any* neighboring[1] *sequences of datasets D, D', and for any outcome set $F \subseteq \mathbb{R}^T$ it holds that $\Pr[\texttt{Releases}_{\mathcal{A}}(D) \in F] \leq e^\varepsilon \cdot \Pr[\texttt{Releases}_{\mathcal{A}}(D') \in F] + \delta$. The case where $\delta = 0$ is referred to as* pure *DP and the case where $\delta > 0$ as* approximate *DP.*

Remark 1.2. *Typically one aims for a stronger privacy definition in which the neighboring datasets are not fixed in advanced, and can be adaptively chosen as the execution progresses. Since in this work we focus on* negative *results, then working with the above "non-adaptive" privacy definition only makes our results stronger.*

An algorithm that operates in the continual observation model must overcome two main challenges: (i) The algorithm is required to solve T instances of the underlying problem P, rather than just a single instance, and (ii) The algorithm receives its input in an online manner, which may introduce larger error than what is required when the algorithm receives the entire sequence at once. So far, all known hardness results for the continual observation model were based on challenge (i) above. To emphasize this, let us consider the following "offline" variant of the continual observation model:

Definition 1.3 (The offline observation model)**.** *Let $P : X^* \to \mathbb{R}$ be a problem. In the* offline observation model*, the algorithm gets in the beginning of the execution an entire sequence of datasets $D = (D_1, \ldots, D_T)$, and aims to return T solutions $y_1, ..., y_T$ such that $y_t \approx P(D_1 \circ \cdots \circ D_t)$ for every $t \in [T]$.*

So far, all prior hardness results for the continual observation model are, in fact, hardness results for this offline observation model. On the one hand, this makes these prior results stronger, as the

[1] $D = (D_1, \ldots, D_T)$ and $D' = (D'_1, \ldots, D'_T)$ are neighboring if $D_t = D'_t$ for all but at most index t^* for which D_{t^*} could be obtained from D'_{t^*} by adding/removing one element.

offline observation model is strictly easier than the continual observation model. On the other hand, such lower bounds cannot capture the hardness that steams from the online nature of the continual observation model. As we show in this work, there are important cases where in order to show strong lower bounds for the continual observation model we *must* differentiate it from the offline observation model.[2] Furthermore, we show this for what is arguably the most well-studied problem in the continual observation model – the *private counter* problem. In this problem, the data domain is $X = \{0, 1\}$ and the function P simply counts the number of ones in the datasets received so far. Alternatively, it is common to interpret the input as a sequence of non-negative numbers $\Delta = (\Delta_1, \ldots, \Delta_T) \in \mathbb{Z}^T_{\geq 0}$, where Δ_t is the number of *events* that occurred at time t, and the goal is to track (and publish) the prefix sums $n_t = \sum_{i=1}^{t} \Delta_i$ for $t \in [T]$.

1.1 Prior Works on the Counter Problem

Upper bounds. Dwork et al. (2010a) and Chan et al. (2010) presented the celebrated *binary tree mechanism* for the online counter problem, which guarantees error at most $O_\varepsilon(\log(T) + \log^{2.5} n)$, where $n = \sum_{t=1}^{T} \Delta_t$ is the total number of events throughout the execution (via the analysis of Dwork et al. (2015)). In the offline observation model, the private counter problem can be solved with error $\tilde{O}_{\varepsilon,\delta}((\log^* T)(\text{polylog}(n)))$ via algorithms for the related problem of *threshold sanitization* (a.k.a. CDF estimation) (Cohen et al., 2023; Kaplan et al., 2020). So, in terms of the dependence in the time horizon T, the private counter problem can be solved with error $O(\log T)$ in the online model and $\tilde{O}(\log^* T)$ in the offline model.

Lower bounds. There are three prior lower bounds for the private counter problem, all of which hold also for the offline model:

(1) Dwork et al. (2010a) showed a lower bound of $\Omega(\log T)$ that holds only for *pure*-DP algorithms.

(2) For approximate DP, a lower bound of $\Omega(\log^* T)$ follows from the results of Bun et al. (2015), who showed a lower bound for the simpler *private median* problem. Thus, in the offline setting, the dependence on T of $\tilde{\Theta}(\log^* T)$ is tight.

(3) Henzinger et al. (2023) recently established a lower bound of $\Omega(\min\{\log T, \log n\})$ on the ℓ_2 error, which implies a lower bound of $\Omega(\min\{\log T, \log n\})$ on the ℓ_∞ error.

Our focus in this work is on lower bounds on the ℓ_∞ error of approximate-DP online counters. To summarize our knowledge from prior works: The known upper bound is $O(\min\{n, \log T + \log^2 n\})$, combining the bound of Dwork et al. (2015) with the observation that we can trivially obtain an error of n by reporting $\tilde{n}_t = 0$ for $t \in [T]$). The prior lower bound is $\Omega(\min\{\log T, \log n\})$ by Henzinger et al. (2023). Note that there is a gap: In the common regime where $n \leq T$ the upper bound is $O(\log T + \log^2 n)$ while the lower bound is only $\Omega(\log n)$. Furthermore, as the lower bound of Henzinger et al. (2023) applies in the offline setting, its dependence on T cannot be tightened, since the offline setting has an upper bound of $\tilde{O}((\log^* T)\text{polylog}(n))$.

Remark 1.4. *Most applications for the counter problem operate in the sparse regime where $n \ll T$. For example, in tracking the view counts of YouTube videos, popular videos like those of Taylor Swift may experience rapid increases in views, while the majority of videos tend to exhibit slower growth*

[2] We remark that in other models of computation there are known hardness result that differentiate between similar offline and online models, most notably by Bun et al. (2017).

(that is, small n). This typical pattern is broadly known as the long-tail phenomena. In this use case, a counter is maintained for each video. We may need frequent refreshes (e.g. large T), for example, so that we can detect sudden surges quickly and verify the content for compliance.

1.2 Our Contributions

We present a lower bound of $\Omega(\min\{n, \log T\})$ on the ℓ_∞ error of approximate-DP online counting. This bound is tight in terms of the dependence on T and is tight overall for sparse inputs, where $\log^2 n = O(\log T)$. Formally,

Theorem 1.5. *For any $T \geq 1$, let $n \leq \frac{1}{2}\log(T)$ and $\delta \leq \frac{1}{100n}$. Any $(0.1, \delta)$-DP algorithm for the online counter problem with n events over T time steps must incur ℓ_∞-error of $\Omega(n)$ with constant probability.*

Observe that this gives a lower bound of $\Omega(\log T)$ even when n is as small as $\log T$. As we explained, such a lower bound does not hold for the offline model, and hence our result differentiates between these two models.

The threshold monitor problem. To obtain our results, we consider a simpler variant of the counter problem, which we call the *Threshold Monitor Problem*, which is natural and of independent interest. Similarly to the counter problem, on every time step t we obtain an input Δ_t indicating the number of events occurring on step t. However, instead of maintaining a counter for the total number of events, all we need to do is to indicate when it (roughly) crosses some predefined threshold k. Formally,

Problem 1.6 (The threshold monitor problem)**.** *There is a pre-determined threshold value k. The input to the algorithm is a sequence of T updates $(\Delta_1, \ldots, \Delta_T) \in \{0,1\}^T$. At each time step $t \in [T]$, the algorithm receives the update Δ_t, lets $n_t \leftarrow \sum_{j=1}^{t} \Delta_j$, and outputs $\top$ or $\perp$, subject to the following requirements:*

- *If $n_t < \frac{1}{2}k$, the algorithm reports $\perp$.*
- *If $n_t \geq k$, the algorithm reports $\top$.*
- *If $n_t \in [\frac{1}{2}k, k)$, the algorithm may report either $\perp$ or $\top$.*
- *Once the algorithm reports a $\top$, it halts.*

We say the algorithm succeeds if all its responses (before halting) satisfy the requirements above.

Clearly, an algorithm for the counter problem with ℓ_∞ error at most $k/2$ can be used to solve the Threshold Monitor Problem. Therefore, a lower bound for the Threshold Monitor Problem implies a lower bound for the counter problem. We establish the following, which implies Theorem 1.5. The proof is included in Section 2.

Theorem 1.7. *Consider the Threshold Monitor Problem with $T \geq 1$, $k \leq \frac{1}{2}\log(T)$, and privacy parameter $\delta \leq \frac{1}{100k}$. There is no $(0.1, \delta)$-DP algorithm that succeeds with probability 0.99.*

A natural question is whether the requirement of $\delta < \frac{1}{\log(T)}$ can be relaxed to $\delta < o(1)$. The following observation suggests that the answer is negative.

Observation 1.8. *There is an (ε, δ)-DP algorithm for the Threshold Monitor Problem with threshold $k = O\left(\min\left\{\frac{\log T}{\varepsilon}, \frac{1}{\delta}\right\}\right)$ that succeeds with constant probability.*

This follows as a special case of the classical `AboveThreshold` algorithm of Dwork et al. (2009), also known as the *Sparse Vector Technique*. The algorithm performs an online sequence of T noisy threshold tests and halts after the first time that the response is that the threshold is exceeded. The overall privacy cost is that of a single query. For the threshold monitor problem, we test sequentially each prefix sum if it is above the threshold $3k/4$. We return $\perp$ on "below" responses and return $\top$ (and halt) on the first "above" response. We require that the ℓ_∞ norm of the noise does not exceed $k/4$ (w.h.p.), which holds when $k = \Omega\left(\frac{\log T}{\varepsilon}\right)$. The $O\left(\frac{1}{\delta}\right)$ upper bound follows by simply releasing every input bit with probability δ, and estimating the sum of the bits from that.

1.2.1 Application I: The threshold queries problem

We show that our lower bound extends to the *threshold queries problem*, thereby solving an open question of Bun et al. (2017). In this problem, we get an input dataset D containing n points from $[0, 1]$. Then, for T rounds, we obtain a query $q \in [0, 1]$ and need to respond with an approximation for the number of points in D that are smaller than q. The privacy requirement is that the sequence of all our answers together satisfies DP w.r.t. D. Note that here the dataset is fixed at the beginning of the execution and it is the *queries* that arrive sequentially.

In the offline variant of this problem, where all the queries are given to us in the beginning of the execution (together with the dataset D), this problem is equivalent to the offline counter problem, and can be solved with error $\tilde{O}(\log^* T)$ using the algorithm of Cohen et al. (2023); Kaplan et al. (2020). For the online variant of the problem (where the queries arrive sequentially), Bun et al. (2017) presented an upper bound of $O(\log T)$, and asked if it can be improved. We answer this question in the negative and show the following theorem.

Theorem 1.9. *Let $n, T \geq 1$ be two integers. There is no $(0.1, \frac{1}{n^2})$-DP algorithm $\mathcal{A}$ that, on input a set S of n reals number from $[0, 1]$, answers a sequence of T online but non-adaptive threshold queries within error $\frac{1}{2}\min\{n, \log(T)\}$ and with probability 0.99.*

1.2.2 Application II: Private online learning

We apply our technique to establish a lower bound for the *private online learning* problem in the *mistake bound model*. To the best of our knowledge this is the first non-trivial lower bound for the private online learning model, separating it from the non-private online learning model. This partially answers a COLT'22 open question posted by Sanyal and Ramponi (2022).[3] Our lower bound also implies, for the first time, a separation between the problems of *private online learning* and *private online prediction*. Let us recall these two related models.

The private online learning model. Suppose $\mathcal{X}$ is the data domain and Let $\mathcal{H} \subseteq \{0, 1\}^{\mathcal{X}}$ be a hypothesis class. The private online learning model, considered first by Golowich and Livni (2021), can be expressed as a T-round game between an *adversary* $\mathcal{B}$ and a learning algorithm (the learner) $\mathcal{A}$. In every round i of this game:

[3] This does not completely resolves their question as they aimed for a stronger separation.

1. The adversary chooses a query $x_i \in \mathcal{X}$ and the learner chooses a hypothesis $h_i : \mathcal{X} \to \{0,1\}$.
2. At the same time, the adversary gets h_i and the learner gets x_i.
3. The adversary chooses a "true" label y_i under the restriction that $\exists h \in \mathcal{H}$ such that $\forall j \leq i$ we have $h(x_j) = y_j$. This "true" label y_i is given to the learner.

We say that a mistake happened in round i if $h_i(x_i) \neq y_i$. The goal of the learner is to minimize the total number of mistakes. The privacy definition is that the sequence of all hypotheses $(h_1, h_2, \ldots, h_T)$ together satisfies differential privacy w.r.t. the inputs given to the algorithm throughout the execution. That is, if we were to replace one labeled example (x_i, y_i) with another (x_i', y_i') then that should not change by much the distribution on the resulting vector of hypotheses.

Remark 1.10. *In Step 1 of the game above, the adversary and the learner chooses the next query and hypothesis* simultaneously*. There is another formulation of the model in which the learner is allowed to choose the current hypothesis h_i as a function of the current query x_i. Note that, in both models, $h_i(x_i)$ clearly must depend on x_i. The question is if the description of the hypothesis itself is allowed to depend on x_i (in a way that preserves DP). We do not know if our lower bound holds also for this alternative formulation of the model.*

The private online prediction model. This is a similar model, introduced by Kaplan et al. (2023), which incorporates the following two changes: First, in Step 2 of the above game, instead of getting the hypothesis h_i, the adversary only gets $h_i(x_i)$. Second, the privacy requirement is weaker. Specifically, instead of requiring that the vector of *all* outputs given by the learner should be insensitive to changing one labeled example (x_i, y_i), the requirement is only w.r.t. the vector of all outputs *other than the output in round i*. Intuitively, in the context of private online prediction, this privacy definition allows the ith prediction to depend strongly on the ith input, but only very weakly on all other inputs. This privacy notion could be a good fit, for example, when using a (privacy preserving) online learner to predict a medical condition given an individual's medical history: As long as the ith individual does not share its medical data with additional users, only with the learner, then its medical data will not leak to the other users via the predictions they receive. Kaplan et al. (2023) showed strong positive results that outperform the known constructions for the standard model of private online learning. However, before our work, it was conceivable that the two models are equivalent, and no separation was known.

We consider the class of *point functions* over a domain $\mathcal{X}$, containing all functions that evaluate to 1 on exactly one point of the domain $\mathcal{X}$.[4] This class is known to have Littlestone dimension 1, and hence there is a non-private online-learner for it that makes at most 1 mistake throughout the execution (no matter what T is). We apply our techniques to establish a non-trivial lower bound for the private online learning model, thereby separating it from the non-private model. This is captured by the following theorem.

Theorem 1.11. *Let $T \geq 100$ denote the number of time steps, and let $\mathcal{X} = \{0, 1, 2, \ldots, T\}$ be the data domain. Let $\mathcal{H}$ be the class of* point functions *over $\mathcal{X}$. Let $\delta = \frac{1}{20 \log(T)}$. Then, any (ε, δ)-DP online learner for $\mathcal{X}$ that answers T queries must make $\Omega(\log(T))$ mistakes with constant probability.*

[4] That is, for every $x \in \mathcal{X}$, the class of point functions contains the function c_x defined as $c_x(y) = 1$ iff $y = x$.

We complement this result with an upper bound for the online private prediction model that achieves a mistake bound which is independent of T. This establishes a separation between private online learning and online prediction. Formally,

Lemma 1.12. *Let $\mathcal{X}$ be a domain. The class of point functions over $\mathcal{X}$ admits an (ε, δ)-DP online predictor with mistake bound $O(\frac{1}{\varepsilon}\log\frac{1}{\delta})$.*

The proofs of Theorem 1.11 and Lemma 1.12 are given in the appendix.

2 A Lower Bound for the Threshold Monitor Problem

In this section we prove our lower bound for the threshold monitor problem (i.e., we prove Theorem 1.7), which directly implies our lower bound for the counter problem. We begin with an overview of the proof.

2.1 Proof Overview

Recall that in the threshold monitor problem, the algorithm receives a sequence of updates $\Delta_1, \ldots, \Delta_T \in \{0,1\}^T$ and needs to report "HALT" at some time step $t \in [T]$ such that $\sum_{i=1}^{t} \Delta_i \in [k/2, k]$, where $k = \frac{1}{2}\log(T)$. We want to show that no $(\varepsilon = 0.1, \delta = \frac{1}{100k})$-DP algorithm can solve the problem with success probability 0.99. Suppose such an algorithm exists and denote it by $\mathcal{A}$. Then, on the sequence $D^{(0)} = (0, 0, \ldots, 0)$, $\mathcal{A}$ cannot halt with probability larger than 0.01.

To derive a contradiction, we want to use the fact that $\mathcal{A}$ is differentially private to find a new data set $D^{(k)} \in \{0,1\}^T$ that contains k "1"s in the sequence, yet the algorithm halts on $D^{(k)}$ with probability at most 0.5. If we just append k "1" at the beginning of the sequence, the DP property of $\mathcal{A}$ only promises that $\mathcal{A}$ halts with probability at most: $0.01e^{\varepsilon k} + \sum_{i=1}^{k} \delta \cdot e^{(k-i)\varepsilon}$, which is a vacous bound for $\delta = \frac{1}{100k}$.

In this naive attempt, we only considered the probability that $\mathcal{A}$ ever halts on the input. By the DP property of the algorithm, the best we can say about this quantity is that it increases by a multiplicative factor of e^{ε} and an additive factor of δ when we move to an adjacent data set. The additive increment is easy to bound. However, to handle the multiplicative blow-up, we need to exploit the online nature of the problem and the long time horizon. Specifically, on an input sequence $S = (\Delta_1, \ldots, \Delta_T) \in \{0,1\}^T$, we consider whether $\mathcal{A}$ is more likely to halt on the stream's first or second half.

- If $\mathcal{A}$ more likely halts at time step $t \in [1, T/2]$, then we fix the first half of S, and recursively insert more "1"s in the second half of S. Because $\mathcal{A}$ is an online algorithm, only the second half of the stream will suffer from the "e^{ε}" blow-up in the future.
- If $\mathcal{A}$ more likely halts at time step $t \in [T/2+1, T]$, then we fix the second half of S, and recursively insert more 1's in the first half of S. In doing so, the behavior of $\mathcal{A}$ in time interval $[T/2+1, T]$ might change. However, our construction will eventually append at least k "1"s before time step $T/2$. Therefore, ultimately, the algorithm's behavior on the second half of the stream is irrelevant.

Using this construction, we can start at $D^{(0)} = (0, \ldots, 0)^T$, find a sequence $D^{(k)}$ as well as an index ℓ, such that $D^{(k)}$ contains at least k "1"s before time step ℓ, yet $\mathcal{A}$ halts before time step ℓ with

probability at most 0.5. The key insight in the construction is intuitively explained as follows: after inserting each new item, we cut the stream into two halves and recurse into the left or right half, in a way that at most half of the probability mass will suffer from the e^ε blow-up in the future. Overall, the e^ε blow-up is reduced to a $\frac{1}{2}e^\varepsilon$ multiplicative factor, which is less than 1, and the overall halting probability is easily bounded.

2.2 The Formal Details

We now present the formal construction. We prove the following restatement of Theorem 1.7.

Theorem 2.1. *Let $T > 1$ and $k \le \frac{1}{2}\log(T) - 5$. There is no $(\frac{1}{2}, \frac{1}{100 \cdot k})$-DP algorithm for the Threshold Monitor Problem (Problem 1.6) with parameters T, k that has success probability 0.99.*

Proof. Fix k and let $T = \sum_{i=1}^{k} 2^i$. It is clear that $k \le \frac{1}{2}\log(T) - 5$. We show that there is no algorithm for the Threshold Monitor Problem with parameters k and T (or more) updates. We assume that such an algorithm $\mathcal{A}$ exists and derive a contradiction in the following.

For any fixed input sequence $D \in \{0,1\}^T$ and $t \in [T]$, an algorithm $\mathcal{A}$ for Problem 1.6 can be fully described by a probability distribution on $[T] \cup \{\text{non}\}$ that corresponds to the time step $t \in [T]$ where it outputs $\top$ and halts. The event non corresponds to the algorithm outputting $\perp$ on all time steps. We use the notation p_t^D for the probability that $\mathcal{A}$ outputs $\top$ (and thus halts) after the t-th update:

$$p_i^D := \Pr[\mathcal{A}(D) \text{ outputs } \top \text{ after the } i\text{-th update}]. \tag{1}$$

We start with the all-zero sequence $D^{(0)} = (0, 0, \dots)$ and construct a "hard sequence" for algorithm $\mathcal{A}$, as described in Algorithm 1.

Algorithm 1: Construction of Hard Instance

Input: Parameters k, $T = \sum_{j=1}^{k} 2^j$; A mapping p_i^D as in (1) induced by an Algorithm $\mathcal{A} : \{0,1\}^T$.

Initialize $\ell \leftarrow 1$, $r \leftarrow T$, $D^{(0)} \leftarrow (0, 0, \dots, 0)$, and $c_{\text{total}} \leftarrow 0$
for $i = 1, \dots, k$ **do**
  $m \leftarrow \frac{1}{2}(\ell + r + 1)$ `// midpoint` m `is always an integer`
  $c_{\text{left}} \leftarrow \sum_{j=\ell}^{m-1} p_j^{D^{(i-1)}}$
  $c_{\text{right}} \leftarrow \sum_{j=m}^{r} p_j^{D^{(i-1)}}$
  if $c_{\text{left}} \le c_{\text{right}}$ **then**
    Set the ℓ-th entry of $D^{(i-1)}$ to 1 to obtain $D^{(i)}$
    $c_{\text{total}} \leftarrow c_{\text{total}} + p_\ell^{D^{(i)}}$
    $\ell \leftarrow \ell + 1$
    $r \leftarrow m - 1$
  else
    Set the m-th entry of $D^{(i-1)}$ to 1 to obtain $D^{(i)}$
    $c_{\text{total}} \leftarrow c_{\text{total}} + c_{\text{left}} + p_p^{D^{(i)}}$
    $\ell \leftarrow m + 1$
return ℓ, c_{total} *and* $D^{(k)}$

Note that $D^{(k)}$ is an input sequence with k "1"s before the ℓ-th update. Thus, a correct algorithm has to output $\top$ before the ℓ-th update arrives. However, we will show that, assuming

the algorithm is differentially private and is correct on the first sequence $D^{(0)}$ with probability 0.99, the algorithm must err on the input $D^{(k)}$ with probability at least $\frac{1}{2}$, contradicting to the correctness of the algorithm. We now give the proof. We start with the following simple observation.

Claim 2.2. *Consider Algorithm 1. After the algorithm finishes, it holds that*

$$c_{\text{total}} = \Pr[\mathcal{A}(D^{(k)}) \text{ outputs } \top \text{ before or at the } \ell\text{-th update}].$$

Proof. By definition, we have

$$\Pr[\mathcal{A}(D^{(k)}) \text{ outputs } \top \text{ before or at the } \ell\text{-th update}] = \sum_{i=j}^{r} p_j^{D^{(k)}}.$$

On the other hand, considering the execution of Algorithm 1, we observe that $c_{\text{total}} = \sum_{j=1}^{\ell} p_j^{D^{(k_j)}}$, where we use k_j to denote the round in which the interval $[\ell, r]$ in Algorithm 1 updates to a new interval $[\ell', r']$ with $\ell' > j$. Observe that $p_j^{D^{(k_j)}} = p_j^{D^{(k)}}$ for every j. (This follows because after the k_j-th round, we only update the ℓ'-th to the r'-th entries of the input, and the algorithm is ignorant to future updates at time j.) The claim now follows. □

The next claim gives an upper bound for c_{total}.

Claim 2.3. *Consider Algorithm 1. After the execution finishes, we have* $c_{\text{total}} \le \frac{e^{\varepsilon}(0.01+2k\delta)}{1-\frac{1}{2}e^{\varepsilon}}$.

Proof. We shall use a potential argument. Before the i-th round of Algorithm 1, let the current interval be $[\ell, r]$ and define the current *potential* of the algorithm by $\Phi^{(i-1)} = \sum_{j=\ell}^{r} p_j^{D^{(i-1)}}$. Let $c_{\text{total}}^{(i-1)}$ be the value of c_{total} before the i-th round. We will examine (with some foresight) the following quantity $e^{-\varepsilon} c_{\text{total}}^{(i)} + \frac{1}{1-\frac{1}{2}e^{\varepsilon}} \Phi^{(i)}$. First, assuming the algorithm errs with probability at most 0.01 on the input $D^{(0)}$, it follows that

$$e^{-\varepsilon} c_{\text{total}}^{(0)} + \frac{1}{1-\frac{1}{2}e^{\varepsilon}} \Phi^{(0)} \le 0 + \frac{0.01}{1-\frac{1}{2}e^{\varepsilon}}.$$

Consider how c_{total} and Φ change from round $i-1$ to round i. Note that $c_{\text{total}}^{(i)}$ is obtained by adding a subset $I \subset [\ell, r]$ of $p_j^{D^{(i)}}$ to $c_{\text{total}}^{(i-1)}$. Assuming the algorithm is (ε, δ)-DP, we have

$$c_{\text{total}}^{(i)} = c_{\text{total}}^{(i-1)} + \sum_{j \in I} p_j^{D^{(i)}} \le c_{\text{total}}^{(i-1)} + (e^{\varepsilon} \sum_{j \in I} p_j^{D^{(i-1)}} + \delta) \le c_{\text{total}}^{(i-1)} + e^{\varepsilon} \Phi^{(i-1)} + \delta.$$

On the other hand, $\Phi^{(i)}$ consists of contribution from a sub-interval J of $\Phi^{(i-1)}$, where the interval has a total contribution being at most half of $\Phi^{(i-1)}$. Again assuming the algorithm is (ε, δ)-DP, we have $\Phi^{(i)} = \sum_{j \in J} p_j^{D^{(i)}} \le \delta + e^{\varepsilon} \cdot \sum_{j \in J} p_j^{D^{(i-1)}} \le \frac{1}{2} e^{\varepsilon} \Phi^{(i-1)} + \delta$. Thus,

$$\begin{aligned} e^{-\varepsilon} c_{\text{total}}^{(i)} + \frac{1}{1-\frac{1}{2}e^{\varepsilon}} \Phi^{(i)} &\le e^{-\varepsilon} \left(c_{\text{total}}^{(i-1)} + e^{\varepsilon} \cdot \Phi^{i-1} + \delta \right) + \frac{1}{1-\frac{1}{2}e^{\varepsilon}} \left(\frac{1}{2} e^{\varepsilon} \Phi^{(i-1)} + \delta \right) \\ &= \left(e^{-\varepsilon} c_{\text{total}}^{(i-1)} + \frac{1}{1-\frac{1}{2}e^{\varepsilon}} \Phi^{(i-1)} \right) + \frac{2\delta}{1-\frac{1}{2}e^{\varepsilon}}. \end{aligned}$$

Consequently, $e^{-\varepsilon} c^{(k)}_{\text{total}} + \frac{1}{1-\frac{1}{2}e^{\varepsilon}} \Phi^{(k)} \leq \frac{0.01+2k\delta}{1-\frac{1}{2}e^{\varepsilon}}$, which implies the desired conclusion $c^{(k)}_{\text{total}} \leq \frac{e^{\varepsilon}(0.01+2k\delta)}{1-\frac{1}{2}e^{\varepsilon}}$ as Φ is always non-negative. □

For our choice of $\varepsilon = \frac{1}{2}$ and $\delta = \frac{1}{100k}$, we have $\frac{e^{\varepsilon}(0.01+2k\delta)}{1-\frac{1}{2}e^{\varepsilon}} \leq \frac{1}{2}$. Hence, combining the two claims above shows that the algorithm errs on the input $D^{(k)}$ with a probability of at least 0.5, completing the proof. □

3 Online Threshold Queries

Recall that in the *threshold queries problem*, we get an input dataset D containing n points from $[0, 1]$. Then, for T rounds, we obtain a query $q \in [0, 1]$ and need to respond with an approximation for the number of points in D that are smaller than q. In this section we show that our lower bound extend also to this problem. We begin with a proof overview.

3.1 Proof Overview

The key challenge in extending our lower bound to the online threshold query problem is that the data set is given all at once. To motivate, let us consider a simple reduction:

- For an input sequence $(\Delta_1, \ldots, \Delta_T) \in \{0, 1\}^T$ to the counter problem, construct a data set $D = \{\frac{i}{T+1} : \Delta_i = 1\}$. Then, issue a set of T queries $\{q_i = \frac{i}{T}\}_{1 \leq i \leq T}$ to the threshold query algorithm and report the answers to these queries.

For the offline version of the problem, this reduction preserves the privacy parameters and the correctness perfectly. However, this reduction itself does not work in the online setting: at any time step $t \in [1, T]$, the algorithm does not know the future inputs $\Delta_{t+1}, \ldots, \Delta_T$. Therefore, the data set D cannot be constructed at all without seeing the whole stream.

To carry out the reduction for the online version of the problem, we only need one more property for the threshold query algorithm: on a query $q \in [0, 1]$, the algorithm's response only depends on the set $D \cap [0, q]$. Namely, it is ignorant of all the data points with values larger than q. This condition is not necessarily satisfied by an arbitrary algorithm. Still, for any fixed algorithm $\mathcal{A}$, it is possible to identify a set of T points $0 < p_1, \ldots, p_T < 1$ (possibly depending on $\mathcal{A}$), such that the condition *is* satisfied for any data set $D \subseteq \{p_1, \ldots, p_T\}$ that only contains a subset of these T points. The existence of such T points is guaranteed by the hypergraph Ramsey theory. The high-level idea is easy to describe: Knowing the set $S = D \cap [0, q]$, for every possible data set D that contains S, the algorithm's behavior on q can be thought of as a "coloring" to the data set D. Since there are infinitely many points in $(q, 1]$, one can use the Ramsey theory to identify an arbitrarily large monochromatic clique among $(q, 1]$. That is, a set of data points $P \subseteq (0, q]$ such that the algorithm behaves similarly on any data set $D = S \cup P'$ with $P' \subseteq P$. Note that this is exactly what our reduction asks for.

3.2 The Formal Details

We are now ready to prove Theorem 1.9. We will do so by proving lower bounds for the following threshold version of the problem.

- The input D consists of k data points, each taking values from $[0, 1]$. The algorithm receives all of the set D in advance.
- The algorithm then processes a sequence of queries $q_1 \le q_2 \le \cdots \le q_T$ where each query $q_i \in [0, 1]$ is described by a real number. Let $c_i = |\{j : x_j \le q_i\}|$.
  - If $c_i \le \frac{1}{2}k$, the algorithm reports $\perp$.
  - If $c_i \ge k$, the algorithm reports $\top$.
  - If $c_i \in (k/2, k)$, the algorithm may report either $\perp$ or $\top$.
  - Once the algorithm reports a $\top$, it halts.

We say the algorithm succeeds if all the queries before halting are answered correctly. For $k = \frac{1}{2}\log(T)$ and $\varepsilon = 0.1, \delta = \frac{1}{100k}$, we show that no (ε, δ)-DP algorithm can solve the problem with success probability more than 0.99.

3.2.1 Ramsey Theory

We need a well-known result from Ramsey theory. Consider the complete k-uniform hypergraph over vertex set V (that is to say, for every subset of size k, there is a hyperedge connecting them). For any integer $c \ge 1$, a c-coloring of the graph is a mapping $f : \binom{V}{k} \to [c]$ which assigns a color to each hyperedge. Given a mapping f, a subset $S \subseteq V$ of vertices is called a *monochromatic clique* if all hyperedges inside S are colored using the same color. Then, the hypergraph Ramsey theorem says the following.

Lemma 3.1. *Fix any finite $k, c, n \ge 1$. Then, there is a sufficiently large N such that, for every c-coloring to a complete k-uniform hypergraph with at least N vertices, a monochromatic clique of size at least n exists.*

3.2.2 The Key Lemma

We need the following technical lemma for the reduction to go through.

Lemma 3.2. *Let $\mathcal{A}$ be an algorithm for the online threshold problem that operates on k data points. Then, for any $\eta > 0$, there exists a subset $0 < x_1 < \cdots < x_T$ of T points satisfying the following. For any fixed data set $S \subseteq \{x_1, \ldots, x_T\}, |S| = k$ and the sequence of queries $x_1, \ldots, x_T$, there exists a sequence of real numbers $r_1 \le r_2 \le \cdots \le r_T$, for which the following holds.*

- *For every $i \in [T]$, $\left|r_i - \Pr[\mathcal{A}(S)$ outputs $\top$ for the i-th query $\mid \mathcal{A}(S)$ did not halt before$]\right| \le \eta$.*
- *r_i depends only on the set $S \cap [0, x_i]$. Namely, it is oblivious to data points larger than x_i.*

Proof. Let N be a large enough integer to be specified later. We construct x_i, p_i in the increasing order of i. To start, consider the discrete ensemble of points $P = \{p_i = \frac{i}{N+1}\}_{1 \le i \le N} \subseteq [0, 1]$.

We set $x_1 := p_1$. Then, consider the k-uniform complete hypergraph with vertex set $V = P \setminus \{p_1\}$. Let $S \subseteq V$ be a hyperedge over k vertices. We color the edge with the color

$$\left\lfloor \frac{1}{\eta} \cdot \Pr[A(S) \text{ outputs } \top \text{ for the first query}] \right\rfloor \in [0, \frac{1}{\eta}].$$

As the number of colors is finite, for any desired $N^{(1)} > 1$, there exists N such that there exists a $N^{(1)}$-monochromatic clique. Denote by $P^{(1)} \subseteq P \setminus \{p_1\}$ the subset.

Next, we consider a $(k-1)$-uniform complete hypergraph with vertex set $P^{(1)}$. Let $S \subseteq P^{(1)}$ be a hyperedge over $(k-1)$ vertices. Color the edge with the color

$$\left\lfloor \frac{1}{\eta} \cdot \Pr[A(S \cup \{x_1\}) \text{ outputs } \top \text{ for the first query}] \right\rfloor.$$

Again, for any desired $N^{(2)}$, there exists $N^{(1)}$ so that the existence of an $N^{(2)}$-monochromatic clique is guaranteed. Denote by $P^{(2)} \subseteq P^{(1)}$ the subset.

Now, note that if we choose x_1 to be p_1 and choose $x_2, \ldots, x_T$ from $P^{(2)}$, then the two conditions in the theorem statement are satisfied for $i = 1$. We proceed to choose $x_2, \ldots, x_T$ from $P^{(2)}$ so that the theorem is also satisfied for $i \geq 2$. We simply choose x_2 as the smallest element in $P^{(2)}$. Then, we apply Ramsey's theory four times, as follows:

- First, We identify a subset $P^{(3)}$ of $P^{(2)}$ such that $\Pr[\mathcal{A}(S \cup \{x_1, x_2\}) \text{ outputs } \top \text{ for the 2nd query }]$ is almost the same (i.e., up to an error of η) for every $S \subseteq P^{(2)}$.
- Continuing, we apply Ramsey's theory three more times to identify $P^{(6)} \subseteq P^{(5)} \subseteq P^{(4)} \subseteq P^{(3)}$ such that, for every data set $S \subseteq P^{(6)} \cup \{x_1, x_2\}$, the quantity

  $$\Pr[\mathcal{A}(S) \text{ outputs } \top \text{ for the 2nd query }]$$

  only depends on $S \cap \{x_1, x_2\}$ up to an error of η.

  In total, we need to apply Ramsey's theory four times because the intersection $S \cap \{x_1, x_2\}$ for a data set $S \subseteq P^{(6)} \cup \{x_1, x_2\}$ has four possibilities.

After this step, we have identified x_1, x_2 and a set $P^{(6)}$ such that the theorem statements are satisfied for $i = 1, 2$ as long as we choose $x_3, \ldots, x_T$ from $P^{(6)}$. By setting N to be large enough, we can make $P^{(6)}$ arbitrarily large. We apply the same idea in the following stages. Namely, at each stage $i \in [T]$, we choose x_i as the smallest element in the current set of "active points" $P^{(t_i)}$. Then, we apply Ramsey's theory enough times to identify $P^{(t_{i+1})}$ such that the theorem is satisfied for the i-th query as long as we choose $x_{i+1} \ldots, x_T$ from $P^{(t_{i+1})}$. Once we have chosen x_T, the construction is complete.

If we start with a sufficiently large N and point set $P = \{\frac{i}{N+1}\}_{i \in [N]}$, we can guarantee the success of each application of Ramsey's theory. This completes the proof. □

3.2.3 The Reduction

We are ready to finish the reduction.

Theorem 3.3. *Let $\mathcal{A}$ be an (ε, δ)-DP algorithm for the online threshold algorithm that operates on k data points and has failure probability at most β. For every $\eta > 0$, let $\{x_1, \ldots, x_T\} \subseteq [0, 1]$ be as given in Lemma 3.2 with parameter η. Then, there exists an algorithm $\mathcal{B}$ for the online Counter Problem that is $(\varepsilon, \delta + T\eta)$-DP and has failure probability $T\eta + \beta$.*

Note that the lower bound for the threshold query problem (i.e., Theorem 1.9) follows by combining Theorem 3.3 and Theorem 1.7.

Proof. We describe the algorithm $\mathcal{B}$.

- $\mathcal{B}$ maintains a subset $S \subseteq \{x_1, \ldots, x_T\}$, which we think of as the "input" to $\mathcal{A}$.
- For each $i \in [T]$, $\mathcal{B}$ receives the i-th update Δ_i. If $\Delta_i = 1$, $\mathcal{B}$ inserts x_i to S. Otherwise, $\mathcal{B}$ does not update S in this round. Then, $\mathcal{B}$ invokes $\mathcal{A}$ with the partial data set S and query x_i. By Theorem 3.2, knowing only $S \subseteq \{x_1, \ldots, x_i\}$, there is $r_i \in [0, 1]$ that captures the halting probability of $\mathcal{A}$ up to an error of η. $\mathcal{B}$ simply returns $\top$ with probability r_i. This finishes the description of the algorithm $\mathcal{B}$.

For the correctness, note that for any update sequence $\Delta = (\Delta_1, \ldots, \Delta_T)$ of Hamming weight at most k, it induces a data set $S = \{x_i : \Delta_i = 1\}$. Furthermore, the output distribution of $\mathcal{B}$ on Δ, and the output of $\mathcal{A}$ on S (and queries $x_1, \ldots, x_T$) differ by at most $\eta \cdot T$ in statistical distance. Furthermore, the correctness criteria of the two problems are equivalent. Hence, if $\mathcal{A}$ has failure probability β, we know that $\mathcal{B}$ has failure probability at most $\eta \cdot T + \beta$.

For the privacy claim, simply note that two adjacent update sequences Δ, Δ' induce two adjacent data sets S, S'. Thus, the privacy of $\mathcal{B}$ follows from the privacy of $\mathcal{A}$ and the similarity of output distributions between $\mathcal{A}$ and $\mathcal{B}$. □

References

Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. Differentially private release and learning of threshold functions. In *FOCS*, 2015.

Mark Bun, Kobbi Nissim, and Uri Stemmer. Simultaneous private learning of multiple concepts. In *ITCS*, 2016.

Mark Bun, Thomas Steinke, and Jonathan R. Ullman. Make up your mind: The price of online queries in differential privacy. In *SODA*, 2017.

T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. In *ICALP*, 2010.

Edith Cohen, Ofir Geri, Tamas Sarlos, and Uri Stemmer. Differentially private weighted sampling. In *AISTATS*, 2021.

Edith Cohen, Xin Lyu, Jelani Nelson, Tamás Sarlós, and Uri Stemmer. $\tilde{\text{O}}$ptimal differentially private learning of thresholds and quasi-concave optimization. In *STOC*, 2023.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC*, 2009.

Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *STOC*, 2010a.

Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *FOCS*, 2010b.

Cynthia Dwork, Moni Naor, Omer Reingold, and Guy N. Rothblum. Pure differential privacy for rectangle queries via private partitions. In *ASIACRYPT*, 2015.

Noah Golowich and Roi Livni. Littlestone classes are privately online learnable. In *NeurIPS*, 2021.

Monika Henzinger, Jalaj Upadhyay, and Sarvagya Upadhyay. Almost tight error bounds on differentially private continual counting. In *SODA*, 2023.

Svante Janson. Tail bounds for sums of geometric and exponential variables. *Statistics & Probability Letters*, 135:1–6, 2018.

Haim Kaplan, Katrina Ligett, Yishay Mansour, Moni Naor, and Uri Stemmer. Privately learning thresholds: Closing the exponential gap. In *COLT*, 2020.

Haim Kaplan, Yishay Mansour, Shay Moran, Kobbi Nissim, and Uri Stemmer. Black-box differential privacy for interactive ml. In *NeurIPS*, 2023.

Michael J. Kearns, Mallesh M. Pai, Ryan M. Rogers, Aaron Roth, and Jonathan R. Ullman. Robust mediators in large games. *CoRR*, abs/1512.02698, 2015.

Amartya Sanyal and Giorgia Ramponi. Do you pay for privacy in online learning? In *Conference on Learning Theory*, pages 5633–5637. PMLR, 2022.

A Lower Bound for Private Online Learning

In this section we prove Theorem 1.11. Recall that this theorem states that every private online learner for the class of all point functions over a somain $\mathcal{X}$ must make $\Omega(\log T)$ mistakes with constant probability. We first give a proof sketch for a simplified lower bound that holds only for *proper* learners, i.e., for learners that on every time step i choose a hypothesis h_i that is itself a point function. For simplicity, in this intuitive explanation we will assume that $|\mathcal{X}| \gg T$, say $|\mathcal{X}| = T^2$. Suppose towards contradiction that there exists a private online *proper* learner $\mathcal{A}$ for the class of all point functions with error $k \ll \log T$. We now use $\mathcal{A}$ in order to construct an algorithm $\mathcal{M}$ that solves the threshold monitor problem (Problem 1.6), which will contradict Theorem 1.7. At the beginning of the execution, algorithm $\mathcal{M}$ chooses a random point $x^* \in \mathcal{X} \setminus 0$. Then, on every time step i we do the following:

1. Algorithm $\mathcal{M}$ obtains an input bit $b_i \in \{0, 1\}$.
2. If $b_i = 0$ then set $x_i = 0$. Otherwise set $x_i = x^*$.
3. Feed the input x_i to algorithm $\mathcal{A}$, obtain a hypothesis h_i, and then feed it the "true" label b_i.
4. If $h_i(x^*) = 0$ then output $\perp$ and continue to the next step. Otherwise output $\top$ and halt.

Now, whenever $\mathcal{A}$ makes at most k mistakes, then algorithm $\mathcal{M}$ halts *before* the $(k+1)$th time in which the input bit is 1. On the other hand, observe that before the first time that we get a positive input bit, then algorithm $\mathcal{A}$ gets no information on x^*, and hence the probability that $h_i(x^*) = 1$ is very small (at most $1/|\mathcal{X}|$). This is because when $\mathcal{A}$ is a proper learner, then h_i can label at most 1 point from $\mathcal{X}$ as 1, and x^* is chosen at random. Hence, when $|\mathcal{X}|$ is large enough, we get that w.h.p. algorithm $\mathcal{M}$ never halts before the first time that we get a positive input bit. This contradicts our impossibility result for the threshold monitor problem.[5]

We now proceed with the formal proof, which is more complex in order to capture also the improper case. Intuitively, the extra difficulty is that in the improper case the algorithm might choose hypotheses that label the point x^* as 1 even before seeing any instance of this point.

Proof. **of Theorem 1.11** Suppose for the sake of contradiction that such a learner exists and denote it by $\mathcal{A}$. Denote $k = \sqrt{T}$, and consider the following distribution over query sequences.

- For the i-th query when $i \in [1, k]$, send $(x_i, 0)$ (where $\mathsf{Uniform}[a, b]$ is the uniform distribution on $\{a, \ldots, b\}$).
- For the ith query when $i \in \{k+1, \ldots, 2k\}$, send $(x_i, 0)$ where $x_i \sim \mathsf{Uniform}[k+1, 2k]$.
- In general for $j \in [k]$, for the ith query where $i \in \{(j-1)k+1, \ldots, jk\}$, send $(x_i, 0)$ where $x_i \sim \mathsf{Uniform}[(j-1)k, jk]$

[5] Technically, in the definition of the threshold monitor problem we required the algorithm to output $\perp$ whenever the number of ones in the stream is at most $k/2$, while here we allowed the algorithm to output $\top$ immediately after the first time that the stream contains a positive input. This is a minor issue which we ignore for the sake of this simplified explanation.

We use k phases of k queries each. For each phase $j \in [k]$ there is an allocated disjoint set of k points $X_j := \{(j-1)k+1, \ldots, jk\}$. The query points at each step of phase j is selected uniformly at random from X_j.

Under this query distribution, for every i, j we denote $p_i^{(j)} := \Pr[h_i(j) = 1]$, where h_i is the hypothesis released at the i-th step by $\mathcal{A}$. Note that the events $\mathbf{1}\{h_i(j) = 1\}$ depend on the randomness of the query sequence and the algorithm and are not necessarily independent.

Note that the query distribution we defined always chooses the label 0. This might seem strange at first glance. What we show, however, is that a private online learner cannot keep releasing the all-zero hypothesis (whereas a non-private learner would naturally do this with this query distribution). That is, a private learner must sometimes "gamble" and output a hypothesis that is not all zero. But these "gambles" (as we show) will inflate the mistake bound of the non-private learner and yield the desired lower bound. Formally, the analysis proceeds by inspecting the following two cases:

Case 1. For every phase $j \in [k]$ and every $k' \in [(j-1)k+1, jk]$, we have

$$\sum_{i=(j-1)k+1}^{jk} p_i^{(k')} \geq 0.001,$$

that is, the average probability, over steps i in the phase, of the probability that the released hypothesis is 1 on k' is at least $0.001/k$.

We can now express the expected number of mistakes by the algorithm:

$$\sum_{j=1}^{k} \sum_{i=(j-1)k+1}^{jk} \frac{1}{k} \sum_{k' \in X_j} p_i^{(k')} \geq \Omega(\sqrt{T}).$$

Case 2. There exists a phase $j \in [k]$ and point $k' \in X_j$ such that

$$\sum_{i=(j-1)k+1}^{jk} p_i^{(k')} \leq 0.001. \tag{2}$$

In this case, we first derandomize the query distribution.

Consider the query sequence and the event that it does not include examples of the form $(k', 0)$, that is, for all steps i in phase j, $x_i \neq k'$. This event has probability $(1 - \frac{1}{k})^k \geq 1/4$. Therefore, from (2) it must hold that when conditioning on the event that k' is not present we have

$$\sum_{i=(j-1)k+1}^{jk} \Pr[h_i(k') = 1] \leq 0.001/0.25 \leq 0.01. \tag{3}$$

This implies that with probability at least 0.99 over sequences conditioned on the event, it holds that $h_i(k') = 0$ for all of the steps $i \in [(j-1)k+1, jk]$.

Therefore we can fix a sequence for which k' is not present and with probability 0.99, $h_i(k') = 0$ on all steps.

We now apply the construction of Algorithm 1 to insert $\Omega(\log(k)) = \Omega(\log(T))$ queries of the form $(k', 1)$ into the sequence, such that the probability that $h_i(k') = 0$ throughout is still $\Omega(1)$. This in particular means that the algorithm makes $\Omega(\log(T))$ mistakes on the new query sequence, as desired. □

Algorithm 2: `JDP-Mirror`

Input: Parameters $\varepsilon < 0.6, \delta$, $K \geq 1$

`// Initialization`

$K_1 \sim P_\delta$; $K_2 \sim P_\delta$ `// independently, where` P_δ `is as in Claim B.2`

$C_1, C_2 \leftarrow 0$ `// counters towards phase transition`

for $t \in [T]$ **do** $r_t \sim \mathsf{Bern}[\varepsilon]$ `// independent step activation bits`

`// Phase I:`

for $t = 1, \ldots$ **do** `// Phase I`

  Receive an input bit $\Delta_t \in \{0, 1\}$

  Output $y_t \equiv \perp$

  if $\Delta_t = 1$ & $r_t = 1$ **then** `// increment counter`

    $C_1 \leftarrow C_1 + 1$

    if $C_1 = K_1$ **then** $t_1 \leftarrow t + 1$; **break** `// test and go to Phase II`

`// Phase II:`

for $t = t_1, \ldots$ **do** `// Phase II`

  Receive an input bit $\Delta_t \in \{0, 1\}$

  if Δ_t & $r_t = 1$ **then**

    Output $y_t = \top$

    $C_2 \leftarrow C_2 + 1$

    if $C_2 = K_2$ **then** $t_2 \leftarrow t + 1$; **break** `// test and go to Phase III`

  else

    Output $y_t = \perp$

`// Phase III:`

for $t = t_2, \ldots$ **do** `// Phase III`

  Receive an input bit $\Delta_t \in \{0, 1\}$

  if $\Delta_t = 1$ **then** Output $y_t = \top$ **else** Output $y_t = \perp$

B JDP Mirror

In this section we present a construction for (a variant of) the threshold monitor problem that satisfies a weaker variant of differential privacy known as *joint differential privacy (JDP)* and has an error that is independent of T. In the following section we will leverage this construction in order to present our private online predictor for point functions. We first recall the definition of JDP.

Definition B.1 (Kearns et al. (2015))**.** *Let $\mathcal{M} : X^n \to Y^n$ be a randomized algorithm that takes a dataset $S \in X^n$ and outputs a vector $\vec{y} \in Y^n$. Algorithm $\mathcal{M}$ satisfies (ε, δ)-joint differential privacy (JDP) if for every $i \in [n]$ and every two datasets $S, S' \in X^n$ differing only on their ith point it holds that $\mathcal{M}(S)_{-i} \approx_{(\varepsilon,\delta)} \mathcal{M}(S')_{-i}$. Here $\mathcal{M}(S)_{-i}$ denotes the (random) vector of length $n-1$ obtained by running $(y_1, \ldots, y_n) \leftarrow \mathcal{M}(S)$ and returning $(y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_n)$.*

We present an algorithm that satisfies JDP for the following `Mirror` problem. For a sequence of input bits $(\Delta_t)_{t \in [T]}$ where $\Delta_t \in \{0, 1\}$, the goal is to mirror the input, returning $y_t = \perp$ for $\Delta_t = 0$ and returning $y_t = \top$ for $\Delta_t = 1$. The constraint is that a $\top$ may be returned only after at least

one prior 1. We permit one-sided mistakes, that is, reporting $y_t = \perp$ when $\Delta_t = 1$. The goal is to minimize the number of mistakes.

Algorithm 2 initializes by sampling independent K_1 and K_2 from the distribution P_δ as in Claim B.2. For input $\Delta_t = 0$, it always returns $y_t = \perp$. For input $\Delta_t = 1$ the response depends on the phase. The algorithm has three internal phases. During the first phase, it returns $y_t = \perp$ for all steps. It maintains a counter C_1 that is incremented by an independent $r_t \sim \mathsf{Bern}[\varepsilon]$ after each input with $\Delta_t = 1$. When $C_1 = K_1$, the algorithm internally transitions to the second phase. During the second phase, when $\Delta_t = 1$ it again samples $r_t \sim \mathsf{Bern}[\varepsilon]$ and increments a counter C_2 by r_2. It also returns $y_t = \top$ when $r_t = 1$. When $C_2 = K_2$, the algorithm internally transitions to the third phase. In the third phase, it always returns $y_t = \top$ when $\Delta_t = 1$.

Claim B.2 (P_δ distribution)**.** *For any $\delta > 0$, there is a discrete distribution P_δ on $\mathbb{N}$ with support $\{1, \ldots, L = O(\log(1/\delta)\}$ and such that $P(1) = P(L) = \delta$, $P(i) \approx_{0.5} P(i+i)$ for $i \in [L-1]$, mean $\mu = (L+1)/2$, and for $c \geq 0$, $\Pr[|X - \mu| > c = e - \Theta(c)$.*

We show the following:

Lemma B.3 (Mistake bound of `JDP-Mirror`)**.** *The expected number of mistakes made by Algorithm 2 is $O(\varepsilon^{-1}\log(1/\delta))$. The probability that the number of mistakes exceed $c \cdot \varepsilon^{-1}\log(1/\delta)$ is $\exp(-c \cdot O(\log(1/\delta)))$.*

Proof. During the first two phases, the number of $\Delta_t = 1$ steps between two that result in incrementing the counter is $\mathsf{Geom}[\varepsilon]$. These $\mathsf{Geom}[\varepsilon]$ random variables are independent. During the first phase we incur mistakes also on the K_1 steps that are counted. During the second phase we do not incur mistakes on the K_2 steps that are counted. The distribution of the number of mistakes conditioned on K_1 and K_2 is therefore $\mathsf{NegBinomial}(K_1 + K_2, \varepsilon) - K_2$.[6] The number of mistakes is stochastically smaller than $\mathsf{NegBinomial}(2L = O(\log(1/\delta), \varepsilon)$ since $K_1 + K_2 = O(\log(1/\delta)$ (by Claim B.2). The expected value is therefore $O(\log(1/\delta/\varepsilon)$ and the concentration claim follows from a bound on the upper tail of the negative binomial distribution (4). □

Theorem B.4 (JDP of `JDP-Mirror`)**.** *Algorithm 2 is $(O(\varepsilon), O(\delta))$-JDP. That is, for any two adjacent sequences that differ in position i^**

$$\boldsymbol{\Delta} = \Delta_1, \ldots, \Delta_{i^*-1}, \Delta_{i^*} = 1, \Delta_{i^*+1}, \ldots, \Delta_T$$
$$\boldsymbol{\Delta}' = \Delta_1, \ldots, \Delta_{i^*-1}, \Delta_{i^*} = 0, \Delta_{i^*+1}, \ldots, \Delta_T$$

and a set Y of potential outputs

$$\Pr[\mathit{JDPM}(\vec{\Delta})_{-i^*} \in Y_{-i^*}] \approx_{O(\varepsilon),O(\delta)} \Pr[\mathit{JDPM}(\vec{\Delta}')_{-i^*} \in Y_{-i^*}].$$

Remark B.5 (`JDP-Delayed-Mirror`)**.** *`JDP-Mirror` (Algorithm 2) as described returns its first $y_t = \top$ after processing $\mathsf{Geom}[\varepsilon]$ tops. `Delayed-Mirror` is an extension of `JDP-Mirror` where for an input parameter $K \geq 1$, the value $K - 1$ is added to the sampled K_1 at initialization. The following holds with this modification:*

- *The first $y_t = \top$ is returned only after $\mathsf{NegBinomial}(K, \varepsilon)$ inputs with $\Delta_t = 1$ are processed. From (5) and (4), for any $c > 1$, the probability of the first "$\top$" after fewer than $K/(c \cdot \varepsilon)$ or more than cK/ε such inputs is $e^{-O(cK)}$.*

[6] Recall that $\mathsf{NegBinomial}(K, \varepsilon)$ is a sum of K $\mathsf{Geom}[\varepsilon]$ random variables.

- *The number of mistakes (extension of Lemma B.3) is stochastically smaller than* $\mathsf{NegBinomial}(K+2L,\varepsilon)$. *Hence the expected number of mistakes is at most* $(K+2L)/\varepsilon$ *and the probability that the number of mistakes exceeds* $c(K+2L)/\varepsilon$ *is* $e^{-O(c(K+2L))}$.

The privacy analysis (Section B.2) goes through with small modifications in the case $i^* < w_1$.

B.1 Notation and tools

Overloading the notation, for two nonnegative reals a, b we write that $a \approx_{\varepsilon,\delta} b$ if $a \le e^{\varepsilon} b + \delta$ and $b \le e^{\varepsilon} a + \delta$. For two random variables X, Y over domain $\mathcal{U}$, we write $X \approx_{\varepsilon,\delta} Y$ if for any $U \subset \mathcal{U}$, $\Pr[X \in U] \approx_{\varepsilon,\delta} \Pr[Y \in U]$. We will interchangeably use notation for random variables and distributions.

We will use the following tail bound for the Negative Binomial distribution Janson (2018): For $c > 1$,

$$\Pr_{X \sim \mathsf{NegBinomial}(n,p)}[X > c \cdot n/p] \le e^{-O(cn)} \tag{4}$$

$$\Pr_{X \sim \mathsf{NegBinomial}(n,p)}[X < \frac{1}{c} \cdot n/p] \le e^{-O(cn)} \ . \tag{5}$$

Privacy Analysis via Excluded Runs We use the following standard tool to simplify out analysis. The tool allows us to establish approximate DP by considering one output at a time, as with pure DP. For a randomized algorithm $\mathcal{M}$ and a predicate F on internal choices or outputs of $\mathcal{M}$ we denote by $F(\mathcal{M}, D)$ the probability of an application of $\mathcal{M}$ to D that satisfies F. For an output y, we denote by $F(\mathcal{M}, D, y)$ the probability that F is satisfied when conditioned on the output being y. For a set Y of potential outputs, we denote by $\Pr[\mathcal{M}(D) \in Y \mid F]$, $\Pr[\mathcal{M}(D) \in Y \mid \not{F}]$ respectively the probability of output in Y conditioned on F holding or not holding.

For two datasets D and D' and y we denote

$$\Pr[\mathcal{M}(D) = y] \approx_{\varepsilon|F} [\mathcal{M}(D') = y]$$

if

$$(1 - F(\mathcal{M}, D, y)) \Pr[\mathcal{M}(D) = y \mid \neg F] \le e^{\varepsilon} \Pr[\mathcal{M}(D') = y]$$
$$(1 - F(\mathcal{M}, D', y)) \Pr[\mathcal{M}(D') = y \mid \neg F] \le e^{\varepsilon} \Pr[\mathcal{M}(D) = y]$$

We use the notation $\mathcal{M}(D) \approx_{\varepsilon|F} \mathcal{M}(D')$ if for all output y it holds that $\Pr[\mathcal{M}(D) = y] \approx_{\varepsilon|F} [\mathcal{M}(D') = y]$.

Lemma B.6 (Excluded Runs)**.** *Let* $\mathcal{M}$ *be a randomized algorithm and* F *be a predicate as above. Let* D *and* D' *be such that*

- $F(\mathcal{M}, D) \le \delta$ *and* $F(\mathcal{M}, D') \le \delta$
- $\mathcal{M}(D) \approx_{\varepsilon|F} \mathcal{M}(D')$

Then $\mathcal{M}(D) \approx_{\varepsilon,\delta} \mathcal{M}(D')$.

Proof. Let Y be a subset of the output domain

$$\begin{aligned}\Pr[\mathcal{M}(D) \in Y] &= \sum_{y \in Y} \Pr[\mathcal{M}(D) = y] \\ &= \sum_{y \in Y} (1 - F(\mathcal{M}, D, y)) \cdot \Pr[\mathcal{M}(D) = y \mid \neg F] + F(\mathcal{M}, D, y) \cdot \Pr[\mathcal{M}(D) = y \mid F] \\ &\leq e^{\varepsilon} \sum_{y \in Y} \Pr[\mathcal{M}(D') = y] + \sum_{y \in Y} F(\mathcal{M}, D, y) \\ &\leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in Y] + \delta \ .\end{aligned}$$

□

B.2 JDP of Algorithm 2

Fix two adjacent sequences:

$$\begin{aligned}\boldsymbol{\Delta} &= \Delta_1, \ldots, \Delta_{i^*-1}, 1, \Delta_{i^*+1}, \ldots, \Delta_T \\ \boldsymbol{\Delta}' &= \Delta_1, \ldots, \Delta_{i^*-1}, 0, \Delta_{i^*+1}, \ldots, \Delta_T\end{aligned}$$

let $\boldsymbol{y}$ and $\boldsymbol{y}'$ respectively be the random variables that are the output of Algorithm 2 on $\boldsymbol{\Delta}$ and $\boldsymbol{\Delta}'$. We need to show that

$$\boldsymbol{y}_{-i^*} \approx_{O(\varepsilon), O(\delta)} \boldsymbol{y}'_{-i^*} \ . \tag{6}$$

Observe that positions $t \neq i^*$ with $\Delta_t = 0$ must have $a_t = \perp$ and moreover, they have no effect on all other outputs algorithm. It therefore suffices to establish (6) on compressed sequences where these positions are removed. Going forward, we assume that $\Delta_t = \Delta'_t = 1$ for $t \neq i^*$ and $\Delta_{i^*} = 1$ and $\Delta'_{i^*} = 0$.

We assume without loss of generality that the input includes at least $2L$ steps t where $\Delta_t = 1$ (and therefore Phase III is reached). When this is not the case we can pad the input and work with a larger T. This assumption simplifies the analysis.

We use the following:

Claim B.7 (Geometric sequence)**.** *For any $\varepsilon < 0.6$, $i \geq 1$ and $X \sim P_\delta$ it holds that $\sum_{j \geq 0} \Pr[X = i + j]\varepsilon^j = O(\Pr[X = i])$*

Proof. It follows from Claim B.2 that for any $j \geq 1$, $\Pr[X = j + 1] \leq e^{0.5} \Pr[X = j]$. For $\varepsilon < 0.6$ we have $\varepsilon e^{0.5} < 1$. □

We specify the following predicate F for excluded runs: F holds when $K_1 \in \{1, L\}$ or when $K_2 \in \{1, L\}$ or when the output (on compressed input) has the general form where for some i, $\boldsymbol{y}_{-i} = \perp^* \top^*$.

Claim B.8 (Specified Excluded Runs)**.** *For any input with at least $2L$ steps (where L is as in Claim B.2) with $\Delta_t = 1$, the probability of predicate F is $O(\delta)$.*

Proof. The events $K_1, K_2 \in \{1, L\}$ have total probability at most 4δ. The probability of an output of the form $\boldsymbol{y}_{-i} = \perp^* \top^*$ is at most that that during Phase II, all the steps with $\Delta_t = 1$ have $y_t = \top$ with the exceptions: When i is in the $\perp$ sequence, then only starting at count $C_2 = 2$. Or when i is in the $\top$ sequence then step $t = i$ is not required to have $y_t = \top$. This probability is for $k \in \{1, 2\}$, at most $\sum_{j \geq k} \Pr[K_2 = i]\varepsilon^{j-1} = O(\delta)$, using Claim B.7. □

We fix an output sequence $\boldsymbol{a}_{-i^*}$ in all positions but i^*. In order to establish (6) it suffices to establish that

$$\Pr[\boldsymbol{y}_{-i^*} = \boldsymbol{a}_{-i^*}] \approx_{O(\varepsilon)|F} \Pr[\boldsymbol{y}'_{-i^*} = \boldsymbol{a}_{-i^*}]. \tag{7}$$

When $\boldsymbol{a}_{-i^*}$ is such that $\Pr[\boldsymbol{y}_{-i^*} = \boldsymbol{a}_{-i^*}] = 0$ and $\Pr[\boldsymbol{y}'_{-i^*} = \boldsymbol{a}_{-i^*}] = 0$ then (7) clearly holds. We therefore assume going forward that this is not the case.

Let

$$w_1 := \arg\min_{t \neq i^*} a_t = \top$$
$$w_2 := \arg\max_{t \neq i^*} a_t = \bot$$
$$A := \sum_{t \in [w_1 : w_2] \setminus \{i_*\}} \mathbf{1}(a_t = \top)$$

respectively, be the position of the first "$\top$" in $\boldsymbol{a}_{-i^*}$, the position of the last "$\bot$" in $\boldsymbol{a}_{-i^*}$, and the number of "$\top$" occurrences in the positions $\{w_1, \ldots, w_2\}$.

Note that there can be at most $2L$ "$\top$" occurrences before w_2. It holds that $w_1 \leq w_2 - 2$ (and equality is possible only when $i^* = w_1 + 1$). It holds that $t_1 \leq w_1$, that is, Phase I must end before $t = w_1$. It also holds that $t_2 > w_2$. That is, Phase III can start only when $t > w_2$. In particular for $w_1 \leq t \leq w_2$ we are at Phase II. The case $A = 0$ corresponds to $w_2 \leq w_1$.

It suffices to establish (7) conditioned on fixing the randomness $\boldsymbol{r}_{-i^*}$ of the activation bits in all time steps $t < w_1$ or $t \neq i^*$. Let $R := \sum_{t \in [w_1 - 1] \setminus \{i^*\}} r_t$ be the number of activated steps before w_1. Note that throughout Phase II, and in particular for $w_1 \leq t \leq w_2$ ($t \neq i^*$), r_t is determined by a_t, that is, $r_t = 1 \iff a_t = \top$. The probability of that is $Q_A = \varepsilon^A$ The remaining randomness in $\boldsymbol{y}_{-i^*}$ we need to consider when computing the probability of output $\boldsymbol{a}_{-i^*}$ is due to r_{i^*}, $r_{w_2+1:T}$, K_1, and K_2.

Case $A = 0$ ($w_2 \leq w_1$) In this case $\boldsymbol{a}_{-i^*}$ has the form $\bot^* \top^*$ and $F \equiv 1$ conditioned on this output. Therefore (7) holds.

Case $A = 1$ ($w_2 > w_1$) With input $\boldsymbol{\Delta'}$ we have $\Delta'_{i^*} = 0$. We must have $K_1 = R$, $C_2(w_2) = A$, and $K_2 > A$. $K_2 = A + j$ is possible only if we get samples of $r_t = 1$ in the first j steps that are not i^* starting from $w_2 + 1$. Hence,

$$\Pr[\boldsymbol{y}'_{-i^*} = \boldsymbol{a}_{-i^*}] = \Pr[K_1 = R] \cdot \varepsilon^A \cdot \sum_{j=1}^{L-A} \varepsilon^j \Pr[K_2 = A + j] \ . \tag{8}$$

With input $\boldsymbol{\Delta}$, we consider the following cases according to where i^* is in the sequence.

- [**case** $i^* > w_2$] It is the same calculation as (8) except that i^* contributes in the last factor, replacing the contribution of the next time step and so forth. There is no difference if the sequence is long enough $T > w_2 + L - A$ as assumed. Therefore (7) holds with equality.
- [**case** $w_1 + 1 \leq i^* \leq w_2 - 1$] We know that step i^* is in Phase II of the algorithm and the first $\top$ was returned after exactly R inputs with $\Delta_t = 1$. Hence, this is only possible with $K_1 = R$. We decompose the probability by conditioning it on the value of the activation r_{i^*}:

$$\Pr[\boldsymbol{y}_{-i^*} = \boldsymbol{a}_{-i^*}] = (1 - \varepsilon) \cdot \Pr[\boldsymbol{y}_{-i^*} = \boldsymbol{a}_{-i^*} \mid r_{i^*} = 0] + \varepsilon \cdot \Pr[\boldsymbol{y}_{-i^*} = \boldsymbol{a}_{-i^*} \mid r_{i^*} = 1] \tag{9}$$

We will establish that

$$\Pr[\boldsymbol{y}_{-i^*} = \boldsymbol{a}_{-i^*} \mid r_{i^*} = 0] = \Pr[\boldsymbol{y}'_{-i^*} = \boldsymbol{a}_{-i^*}] \tag{10}$$
$$\Pr[\boldsymbol{y}_{-i^*} = \boldsymbol{a}_{-i^*} \mid r_{i^*} = 1] \approx_{O(1)|F} \Pr[\boldsymbol{y}'_{-i^*} = \boldsymbol{a}_{-i^*}] \tag{11}$$

Combining (10) and (11) in (9) establishes the claim (7) for this case.

When $r_{i^*} = 0$, C_2 is not incremented in step i^*. The calculation is the same as in (8), which establishes (10).

When $r_{i^*} = 1$ then C_2 is incremented at step i^*. The transition to Phase III still happens at or after step $w_2 - 1$ but at that point we have $C(w_2 - 1) = A + 1$. Therefore,

$$\Pr[\boldsymbol{y}_{-i^*} = \boldsymbol{a}_{-i^*} \mid r_{i^*} = 1] = \Pr[K_1 = R] \cdot \varepsilon^A \cdot \sum_{j=1}^{L-A} \varepsilon^j \Pr[K_2 = A + j + 1] \ . \tag{12}$$

When $A = L - 1$, we exclude the F event of $K_2 = L$. When $A < L - 1$ we apply Claim B.7 to obtain that

$$\sum_{j=1}^{L-A} \varepsilon^j \Pr[K_2 = A + j] = O(\varepsilon \Pr[K_2 = A + 1])$$
$$\sum_{j=1}^{L-A} \varepsilon^j \Pr[K_2 = A + j + 1] = O(\varepsilon \Pr[K_2 = A + 2]) \ .$$

Since $\Pr[K_2 = A + 1] \approx_{O(1)} \Pr[K_2 = A + 2]$ (from Claim B.2) we obtain that

$$\sum_{j=1}^{L-A} \varepsilon^j \Pr[K_2 = A + j + 1] \approx_{O(1)|F} \sum_{j=1}^{L-A} \varepsilon^j \Pr[K_2 = A + j]$$

and therefore this establishes (11).

- **[case $i^* < w_1$]** We follow the same method as the prior step and establish that (10) and (11) hold. When $r_{i^*} = 0$, the calculation is the same as with $\boldsymbol{\Delta}'$, as no counter is incremented, and this establishes (10). When $r_{i^*} = 1$, we consider the following subcases:

  (i) The internal transition I→II occurred before i^*. This is possible only if there are no $r_t = 1$ steps between i^* and w_1 (because then they would be $a_t = \top$). Therefore there are R such steps prior to i^*. It is also not possible for the transition to have occurred prior to the Rth step with $r_t = 1$. This again because then we would have $a_t = \top$. Therefore it must hold that $K_1 = R$. Now there are two possibilities

  (i)A The transition II→III occurred at i^*. This means that $K_2 = 1$, which is in our excluded set F.

  (i)B A transition to Phase III did not occur at i^* and hence we have $C_2 = 2$ at step w_1. We obtain the exact same expression for the conditional probability as (12) and therefore (11) holds for this case as well and the argument proceeds the same way.

(ii) i^* contributed towards the Phase I count and w_1 is the first count of Phase II. Note that this is possible only when $R < L$. In this case, $K_1 = 1 + R$ but otherwise the sequences are "synched" with $C_2 = 1$. We obtain

$$\Pr[\boldsymbol{y}_{-i^*} = \boldsymbol{a}_{-i^*} \mid r_{i^*} = 1] = \Pr[K_1 = R + 1] \cdot \varepsilon^A \cdot \sum_{j=1}^{L-A} \varepsilon^j \Pr[K_2 = A + j] \ . \tag{13}$$

Claim (11) for this case follows using that (Claim B.2) for $1 \leq R < L$, $\Pr[K_1 = R] \approx_{O(1)} \Pr[K_1 = R + 1]$.

C Private Online Prediction for Point Functions

In this section we present a construction for a private online predictor for point functions, obtaining mistake bound that is independent of the time horizon T. This separates private online learning from private online prediction. Specifically, we leverage Algorithm 2 to construct an efficient online learner for point functions in the private prediction model. The construction is described in Algorithm 3.

Lemma C.1. *Let $\mathcal{X}$ be a domain. The class of point functions over $\mathcal{X}$ admits an (ε, δ)-DP online predictor with mistake bound $O(\frac{1}{\varepsilon} \log \frac{1}{\delta})$.*

Proof. The proof is via the construction given in Algorithm 3, which we denote here as algorithm $\mathcal{A}$. The utility analysis of this algorithm is straightforward: Once $\approx \frac{1}{\varepsilon} \log \frac{1}{\delta}$ positive labels have been obtained, we run the private histogram algorithm in order to identify this positively labeled point and the algorithm does not make anymore mistakes from that point forward.

For the privacy analysis, fix two neighboring input streams that differ on their ith labeled example: $S_0 = ((x_1^0, y_1^0), \ldots, (x_T^0, y_T^0))$ and $S_1 = ((x_1^1, y_1^1), \ldots, (x_T^1, y_T^1))$. Let $b \in \{0, 1\}$ and consider the execution of $\mathcal{A}$ on S_b. We now claim that the outcome distribution of $\mathcal{A}(S_b)$, excluding the ith output, can be simulated from the outcomes of three private computations on b. Specifically, we consider a simulator Sim that interacts with a "helping algorithm", Helper, who knows the value of the bit b and answers DP queries w.r.t. b as issued by Sim. The simulator Sim first asks Helper to execute the algorithm for histograms on the dataset containing the first k positively labeled points in S_b to obtain x^*. This satisfies (ε, δ)-DP. The simulator then asks Helper for the outcome vector of executing `JDP-Mirror` on the bit-stream $\left(I(x_j^b, x^*)\right)_{j=1}^T$, where $I(a, b) = 1$ iff $a = b$. The Helper responds with the vector $\vec{b} \in \{\perp, \top, \emptyset\}^T$ (where the ith coordinate is $\emptyset$, denoting the fact that these coordinate is not given to the simulator). Next, similarly to Step 3(c)ii of the algorithm, let $\hat{L}$ denote the indices of the first $10k$ occurrences of x^* in S_b. The simulator asks Helper for the value of

$$\widehat{\text{Fake}} \leftarrow |\{j \in \hat{L} : y_j = 0\}| + \mathsf{Lap}(\frac{1}{\varepsilon}).$$

This satisfies $(\varepsilon, 0)$-DP by the properties of the Laplace mechanism.

We now claim that using these three random elements (x^*, the outcome vector of `JDP-Mirror`, and $\widehat{\text{Fake}}$), we can simulate the outcome of $\mathcal{A}(S_b)$ up to a statistical distance of at most δ. Specifically, if $\widehat{\text{Fake}} \geq \frac{1}{\varepsilon} \log \frac{1}{\delta}$ then we output the all 0 vector, and otherwise we output the vector given by `JDP-Mirror` (where $\top$ is replaced by 1 and $\perp$ is replaced by 0). To see that this simulates the outcome of $\mathcal{A}(S_b)$, let us consider the following two events:

1. The first $\top$ given by `JDP-Mirror` is such that the number of appearances of x^* before it is at least $10k$. By the properties of `JDP-Mirror`, this event happens with probability at least $1 - \delta$.

2. The noise magnitude in $\widehat{\text{Fake}}$ is at most $\frac{1}{\varepsilon} \log \frac{1}{\delta}$. This event also holds with probability at least $1 - \delta$ by the properties of the Laplace distribution.

Now, whenever these two events hold, then the outcome of the simulator is identical to the outcome of $\mathcal{A}(S_b)$. More specifically, if the algorithm sets Flag $= 1$ in Step 3(b)iiiC, then the outcome of the simulator is syntactically identical to the output of the algorithm. Otherwise (if we set Flag $= 3$ in Step 3(b)iiiC), then there must be at least $9k$ occurrences of $(x^*, 0)$ in S_b before the kth occurrence of $(x^*, 1)$. Hence, in this case we have that $\widehat{\text{Fake}} \geq \frac{1}{\varepsilon} \log \frac{1}{\delta}$. So both the algorithm and the simulator output the all zero vector.

Overall, we showed that by posing 3 queries to Helper, each of which satisfies (ε, δ)-DP, our simulator can simulate the outcome of $\mathcal{A}(S_b)$ up to a statistical distance of at most δ. This concludes the proof. $\square$

Algorithm 3: Online Predictor for Point Functions

1. Instantiate the `Delayed-Mirror` (Remark B.5) variant of Algorithm `JDP-Mirror` with the privacy parameter ε and with the threshold $K = 20\varepsilon k$ for $k = \Theta(\frac{1}{\varepsilon}\log\frac{1}{\delta})$.
2. Set Flag $= 0$ and Count $= 0$.
3. For time $t = 1, 2, \ldots, T$ do:
   (a) Obtain an input point x_t.
   (b) If Flag $= 0$ then:
      i. Output 0 and obtain the "true" label y_t.
      ii. Set Count $\leftarrow$ Count $+ y_t$.
      iii. If Count $\geq k = \Theta(\frac{1}{\varepsilon}\log\frac{1}{\delta})$ then
         A. Run an (ε, δ)-DP algorithm for sparse histograms to identify a point $x^* \in \mathcal{X}$ that appears many times in the multiset $\{x_j : j \in [t] \text{ and } y_j = 1\}$. If the algorithm failed to identify such a point then we assume that $x^* = \star$ for some special symbol $\star \notin \mathcal{X}$. See, e.g., Bun et al. (2016); Cohen et al. (2021).
         B. For $j \in [t]$, feed `JDP-Mirror` the bit $I(x_j, x^*)$, where $I(a, b) = 1$ iff $a = b$.
         C. If all the answers given by `JDP-Mirror` in the previous step were $\perp$ then set Flag $= 1$. Otherwise set Flag $= 3$.
   (c) Else if Flag $= 1$ or Flag $= 2$ then:
      i. Feed `JDP-Mirror` the bit $I(x_t, x^*)$ to obtain $a_t \in \{\top, \perp\}$.
      ii. If Flag $= 1$ and $a_t = \top$ then
         - Let $L \subseteq [t]$, where $|L| \leq 10k$, be the set containing the positions of the *first* $10k$ occurrences of x^* till time t.
         - Let Fake $\leftarrow |\{j \in L : y_j = 0\}| + \mathsf{Lap}(\frac{1}{\varepsilon})$.
         - If Fake $\geq \frac{1}{\varepsilon}\log\frac{1}{\delta}$ then set Flag $= 3$ and output 0.
         - Otherwise set Flag $= 2$ and output the bit $I(a_t, \top)$.
      iii. Else output the bit $I(a_t, \top)$.
      iv. Obtain the "true" label y_t.
   (d) Else if Flag $= 3$ then output 0.