Local Differential Privacy for Evolving Data

Matthew Joseph
Computer and Information Science
University of Pennsylvania
majos@cis.upenn.edu

Aaron Roth
Computer and Information Science
University of Pennsylvania
aaroth@cis.upenn.edu

Jonathan Ullman
Computer and Information Sciences
Northeastern University
jullman@ccs.neu.edu

Bo Waggoner
Computer and Information Science
University of Pennsylvania
bowaggoner@gmail.com

Abstract

There are now several large scale deployments of differential privacy used to collect statistical information about users. However, these deployments periodically recollect the data and recompute the statistics using algorithms designed for a single use. As a result, these systems do not provide meaningful privacy guarantees over long time scales. Moreover, existing techniques to mitigate this effect do not apply in the "local model" of differential privacy that these systems use.

In this paper, we introduce a new technique for local differential privacy that makes it possible to maintain up-to-date statistics over time, with privacy guarantees that degrade only in the number of changes in the underlying distribution rather than the number of collection periods. We use our technique for tracking a changing statistic in the setting where users are partitioned into an unknown collection of groups, and at every time period each user draws a single bit from a common (but changing) group-specific distribution. We also provide an application to frequency and heavy-hitter estimation.

1 Introduction

After over a decade of research, differential privacy [12] is moving from theory to practice, with notable deployments by Google [15, 6], Apple [2], Microsoft [10], and the U.S. Census Bureau [1]. These deployments have revealed gaps between existing theory and the needs of practitioners. For example, the bulk of the differential privacy literature has focused on the *central model*, in which user data is collected by a trusted aggregator who performs and publishes the results of a differentially private computation [11]. However, Google, Apple, and Microsoft have instead chosen to operate in the *local model* [15, 6, 2, 10], where users individually randomize their data on their own devices and send it to a potentially untrusted aggregator for analysis [18]. In addition, the academic literature has largely focused on algorithms for performing one-time computations, like estimating many statistical quantities [7, 22, 16] or training a classifier [18, 9, 4]. Industrial applications, however have focused on tracking statistics about a user population, like the set of most frequently used emojis or words [2]. These statistics evolve over time and so must be re-computed periodically.

Together, the two problems of periodically recomputing a population statistic and operating in the local model pose a challenge. Naïvely repeating a differentially private computation causes the privacy loss to degrade as the square root of the number of recomputations, quickly leading to enormous values of ϵ. This naïve strategy is what is used in practice [15, 6, 2]. As a result, Tang et al. [23] discovered that the privacy parameters guaranteed by Apple's implementation of differentially private data collection

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

can become unreasonably large even in relatively short time periods.[1] Published research on Google and Microsoft's deployments suggests that they encounter similar issues [15, 6, 10].

On inspection the naïve strategy of regular statistical updates seems wasteful as aggregate population statistics don't change very frequently—we expect that the most frequently visited website today will typically be the same as it was yesterday. However, population statistics do eventually change, and if we only recompute them infrequently, then we can be too slow to notice these changes.

The *central model* of differential privacy allows for an elegant solution to this problem. For large classes of statistics, we can use the *sparse vector technique* [13, 22, 16, 11] to repeatedly perform computations on a dataset such that the error required for a fixed privacy level grows not with the number of recomputations, but with the number of times the computation's outcome *changes significantly*. For statistics that are relatively stable over time, this technique dramatically reduces the overall error. Unfortunately, the sparse vector technique provably has no local analogue [18, 24].

In this paper we present a technique that makes it possible to repeatedly recompute a statistic with error that decays with the number of times the statistic changes significantly, rather than the number of times we recompute the current value of the statistic, all while satisfying local differential privacy. This technique allows for tracking of evolving local data in a way that makes it possible to quickly detect changes, at modest cost, so long as those changes are relatively infrequent. Our approach guarantees privacy under any conditions, and obtains good accuracy by leveraging three assumptions: (1) each user's data comes from one of m evolving distributions; (2), these distributions change relatively infrequently; and (3) users collect a certain amount of data during each reporting period, which we term an *epoch*. By varying the lengths of the epochs (for example, collecting reports hourly, daily, or weekly), we can trade off more frequent reports versus improved privacy and accuracy.

1.1 Our Results and Techniques

Although our techniques are rather general, we first focus our attention on the problem of privately estimating the average of bits, with one bit held by each user. This simple problem is widely applicable because most algorithms in the local model have the following structure: on each individual's device, data records are translated into a short bit vector using sketching or hashing techniques. The bits in this vector are perturbed to ensure privacy using a technique called *randomized response*, and the perturbed vector is then sent to a server for analysis. The server collects the perturbed vectors, averages them, and produces a data structure encoding some interesting statistical information about the users as a whole. Thus many algorithms (for example, those based on *statistical queries*) can be implemented using just the simple primitive of estimating the average of bits.

We analyze our algorithm in the following probabilistic model (see Section 3 for a formal description). The population of n users has an unknown partition into subgroups, each of which has size at least L, time proceeds in rounds, and in each round each user samples a private bit independently from their subgroup-specific distribution. The private data for each user consists of the vector of bits sampled across rounds, and our goal is to track the total population mean over time. We require that the estimate be private, and ask for the strong (and widely known) notion of *local differential privacy*—for every user, no matter how other users or the server behave, the distribution of the messages sent by that user should not depend significantly on that user's private data.

To circumvent the limits of local differential privacy, we consider a slightly relaxed estimation guarantee. Specifically, we batch the rounds into T *epochs*, each consisting of ℓ rounds, and aim in each epoch t to estimate p^t, the population-wide mean across the subgroups and rounds of epoch t. Thus, any sufficiently large changes in this mean will be identified after the current epoch completes, which we think of as introducing a small "delay".

Our main result is an algorithm that takes data generated according to our model, guarantees a fixed level of local privacy ε that grows (up to a certain point) with the number of distributional changes rather than the number of epochs, and guarantees that the estimates released at the end of each epoch are accurate up to error that scales sublinearly in $1/\ell$ and only polylogarithmically with the total number of epochs T. Our method improves over the naïve solution of simply recomputing the statistic every epoch – which would lead to either privacy parameter or error that scales linearly with the

[1] Although the value of ϵ that Apple guarantees over the course of say, a week, is not meaningful on its own, Apple does take additional heuristic steps (as does Google) that make it difficult to combine user data from multiple data collections [2, 15, 6]. Thus, they may still provide a strong, if heuristic, privacy guarantee.

number of epochs—and offers a quantifiable way to reason about the interaction of collection times, reporting frequency, and accuracy. We note that one can alternatively phrase our algorithm so as to have a fixed error guarantee, and a *privacy cost* that scales dynamically with the number of times the distribution changes[2].

Theorem 1.1 (Protocol for Bernoulli Means, Informal Version of Theorem 4.3)**.** *In the above model, there is an ε-differentially private local protocol that achieves the following guarantee: with probability at least $1-\delta$, while the total number of elapsed epochs t where some subgroup distribution has changed is fewer than $\varepsilon \cdot \min\left(\frac{L}{\sqrt{n\ln(mT/\delta)}}, \ln(T)\sqrt{\frac{n}{\ell}}\right)$, the protocol outputs estimates $\tilde{p}^t$ where*

$$|\tilde{p}^t - p^t| = O\left(\ln(T)\sqrt{\frac{\ln(nT/\delta)}{\ell}}\right)$$

where L is the smallest subgroup size, n is the number of users, ℓ is the chosen epoch length, and T is the resulting number of epochs.

To interpret the theorem, consider the setting where there is only one subgroup and $L = n$. Then to achieve error α we need, ignoring $\log$ factors, $\ell \geq 1/\alpha^2$ and that fewer than $\varepsilon\alpha\sqrt{n}$ changes have occured. We emphasize that our algorithm satisfies ε-differential privacy *for all inputs* without a distributional assumption—only accuracy relies on distributional assumptions.

Finally, we demonstrate the versatility of our method as a basic building block in the design of locally differentially private algorithms for evolving data by applying it to the well-known *heavy hitters* problem. We do so by implementing a protocol due to [3] on top of our simple primitive. This adapted protocol enables us to efficiently track the evolution of histograms rather than single bits. Given a setting in which each user in each round independently draws an object from a discrete distribution over a dictionary of d elements, we demonstrate how to maintain a *frequency oracle* (a computationally efficient representation of a histogram) for that dictionary with accuracy guarantees that degrade with the number of times the distribution over the dictionary changes, and only polylogarithmically with the number of rounds. We summarize this result below.

Theorem 1.2 (Protocol for Heavy-Hitters, Informal Version of Theorem 5.2)**.** *In the above model, there is an ε-differentially private local protocol that achieves the following guarantee: with probability at least $1-\delta$, while the total number of elapsed epochs t where some subgroup distribution has changed is fewer than $\varepsilon \cdot \min\left(\frac{L}{\sqrt{n\ln(mT/\delta)}}, \ln(T)\sqrt{\frac{n\ln(nT/\delta)}{\ell}}\right)$ the protocol outputs estimate oracles $\hat{f}^t$ such that for all $v \in [d]$*

$$|\hat{f}^t(v) - \mathcal{P}^t(v)| = O\left(\ln(T)\sqrt{\frac{\ln(nT/\delta)}{\ell}} + \sqrt{\frac{\ln(2dnT/\delta)}{n}}\right).$$

where n is the number of users, L is the smallest subgroup size, $\mathcal{P}^t$ is the mean distribution over dictionary elements in epoch t, d is the number of dictionary elements, ℓ is the chosen epoch length, and T is the resulting number of epochs.

1.2 Related Work

The problem of privacy loss for persistent local statistics has been recognized since at least the original work of Erlingsson et al. [15] on RAPPOR (the first large-scale deployment of differential privacy in the local model). Erlingsson et al. [15] offers a heuristic memoization technique that impedes a certain straightforward attack but does not prevent the differential privacy loss from accumulating linearly in the number of times the protocol is run. Ding et al. [10] give a formal analysis of a similar memoization technique, but the resulting guarantee is not differential privacy—instead it is a privacy guarantee that depends on the behavior of other users, and may offer no protection to users with idiosyncratic device usage. In contrast, we give a worst-case differential privacy guarantee.

Our goal of maintaining a persistent statistical estimate is similar in spirit to the model of *privacy under continual observation* Dwork et al. [14]. The canonical problem for differential privacy under

[2]We can achieve a dynamic, data-dependent privacy guarantee using the notion of *ex-post* differential privacy [19], for example by using a so-called privacy odometer [21].

continual observation is to maintain a running count of a stream of bits. However, the problem we study is quite different. In the continual observation model, new users are arriving, while existing users' data does not change. In our model each user receives new information in each round. (Also, we work in the local model, which has not been the focus of the work on continual observation.)

The local model was originally introduced by Kasiviswanathan et al. [18], and the canonical algorithmic task performed in this model has become frequency estimation (and heavy hitters estimation). This problem has been studied in a series of theoretical [17, 3, 5, 8, 2] and practical works [15, 6, 2].

2 Local Differential Privacy

We require that our algorithms satisfy *local differential privacy*. Informally, differential privacy is a property of an algorithm A, and states that the distribution of the output of A is insensitive to changes in one individual user's input. Formally, for every pair of inputs x, x' differing on at most one user's data, and every set of possible outputs Z, $\mathbb{P}[A(x) \in Z] \leq e^{\varepsilon} \cdot \mathbb{P}[A(x') \in Z]$. A locally differentially private algorithm is one in which each user i applies a private algorithm A_i *only to their data*.

Most local protocols are *non-interactive*: each user i sends a single message that is independent of all other messages. Non-interactive protocols can thus be written as $A(x_1, \ldots, x_n) = f(A_1(x_1), \ldots, A_n(x_n))$ for some function f, where each algorithm A_i satisfies ε-differential privacy. Our model requires an *interactive* protocol: each user i sends several messages over time, and these may depend on the messages sent by other users. This necessitates a slightly more complex formalism.

We consider interactive protocols among the n users and an additional center. Each user runs an algorithm A_i (possibly taking a private input x_i) and the central party runs an algorithm C. We let the random variable $\mathsf{tr}(A_1, \ldots, A_n, C)$ denote the *transcript* containing all the messages sent by all of the parties. For a given party i and a set of algorithms A'_{-i}, C', we let $\mathsf{tr}_i(x_i; A'_{-i}, C')$ denote the messages sent by user i in the transcript $\mathsf{tr}(A_i(x_i), A'_{-i}, C')$. As a shorthand we will write $\mathsf{tr}_i(x_i)$, since A'_{-i}, C' will be clear from context. We say that the protocol is locally differentially private if the function $\mathsf{tr}_i(x_i)$ is differentially private for every user i and every (possibly malicious) A'_{-i}, C'.

Definition 2.1. *An interactive protocol $(A_1, \ldots, A_n, C)$ satisfies ε-*local differential privacy *if for every user i, every pair of inputs x_i, x'_i for user i, and every set of algorithms A'_{-i}, C', the resulting algorithm $\mathsf{tr}_i(x_i) = \mathsf{tr}_i(A_i(x_i), A'_{-i}, C')$ is ε-differentially private. That is, for every set of possible outputs Z, $\mathbb{P}[\mathsf{tr}_i(x_i) \in Z] \leq e^{\varepsilon} \cdot \mathbb{P}[\mathsf{tr}_i(x'_i) \in Z]$.*

3 Overview: The THRESH Algorithm

Here we present our main algorithm, THRESH. The algorithmic framework is quite general, but for this high level overview we focus on the simplest setting where the data is Bernoulli. In Section 4 we formally present the algorithm for the Bernoulli case and analyze the algorithm to prove Theorem 1.1.

To explain the algorithm we first recall the distributional model. There are n users, each of whom belongs to a subgroup S_j for some $j \in [m]$; denote user i's subgroup by $g(i)$. There are $R = T\ell$ rounds divided into T epochs of length ℓ, denoted $E^1, \ldots, E^T$. In each round r, each user i receives a private bit $x_i^r \sim \mathsf{Ber}(\mu_{g(i)}^r)$. We define the population-wide mean by $\mu^r = \frac{1}{n}(|S_1|\mu_1^r + \ldots + |S_m|\mu_m^r)$. For each epoch t, we use p^t to denote the average of the Bernoulli means during epoch t, $p^t = \frac{1}{\ell}\sum_{r \in E^t} \mu^r$. After every epoch t, our protocol outputs $\tilde{p}^t$ such that $|p^t - \tilde{p}^t|$ is small.

The goal of THRESH is to maintain some public *global estimate* $\tilde{p}^t$ of p^t. After any epoch t, we can *update* this global estimate $\tilde{p}^t$ using *randomized response*: each user submits some differentially private estimate of the mean of their data, and the center aggregates these responses to obtain $\tilde{p}^t$. The main idea of THRESH is therefore to update the global estimate only when it might become sufficiently inaccurate, and thus take advantage of the possibly small number of changes in the underlying statistic p^t. The challenge is to privately identify when to update the global estimate.

The Voting Protocol. We identify these "update needed" epochs through a *voting protocol*. Users will examine their data and privately publish a vote for whether they believe the global estimate needs to be updated. If enough users vote to update the global estimate, we do so (using randomized

response). The challenge for the voting protocol is that users must use randomization in their voting process, to keep their data private, so we can only detect when a large number of users vote to update.

First, we describe a naïve voting protocol. In each epoch t, each user i computes a binary *vote* a_i^t. This vote is 1 if the user concludes from their own samples that the global estimate $\tilde{p}^{t-1}$ is inaccurate, and 0 otherwise. Each user casts a noisy vote using randomized response accordingly, and if the sum of the noisy votes is large enough then a global update occurs.

The problem with this protocol is that small changes in the underlying mean p^t may cause some users to vote 1 and others to vote 0, and this might continue for an arbitrarily long time without inducing a global update. As a result, each voter "wastes" privacy in every epoch, which is what we wanted to avoid. We resolve this issue by having voters also estimate their *confidence* that a global update needs to occur, and vote proportionally. As a result, voters who have high confidence will lose more privacy per epoch (but the need for a global update will be detected quickly), while voters with low confidence will lose privacy more slowly (but may end up voting for many rounds).

In more detail, each user i decides their confidence level by comparing $|\hat{p}^t - \hat{p}_i^{f(t)}|$—the difference between the local average of their data in the current epoch and their local average the last time a global update occurred—to a small set of discrete *thresholds*. Users with the highest confidence will vote in every epoch, whereas users with lower confidence will only vote in a small subset of the epochs. We construct these thresholds and subsets so that in expectation no user votes in more than a constant number of epochs before a global update occurs, and the amount of privacy each user loses from voting will not grow with the number of epochs required before an update occurs.

4 THRESH: The Bernoulli Case

4.1 The THRESH Algorithm (Bernoulli Case)

We now present pseudocode for the algorithm THRESH, including both the general framework as well as the specific voting and randomized response procedures. We emphasize that the algorithm only touches user data through the subroutines VOTE, and EST, each of which accesses data from a single user in at most two epochs. Thus, it is an online local protocol in which user i's response in epoch t depends only on user i's data from at most two epochs t and t' (and the global information that is viewable to all users). THRESH uses carefully chosen *thresholds* $\tau_b = 2(b+1)\sqrt{\ln(12nT/\delta)/2\ell}$ for $b = -1, 0, \ldots, \lfloor \log(T) \rfloor$ to discretize the confidence of each user; see Section 4.2 for details on this choice.

We begin with a privacy guarantee for THRESH. Our proof uses the standard analysis of the privacy properties of randomized response, combined with the fact that users have a cap on the number of updates that prevents the privacy loss from accumulating. We remark that our privacy proof *does not* depend on distributional assumptions, which are only used for the proof of accuracy. We sketch a proof here. A full proof appears in the Supplement.

Theorem 4.1. *The protocol* THRESH *satisfies ε-local differential privacy (Definition 2.1)*

Proof Sketch: Naïvely applying composition would yield a privacy parameter that scales with T. Instead, we will rely on our defined privacy "caps" c_i^V and c_i^E that limit the number of truthful votes and estimates each user sends. Intuitively, each user sends at most $O(\frac{\varepsilon}{a} + \frac{\varepsilon}{b})$ messages that depend on their private data, and the rest are sampled independently of their private data. Thus, we need only bound the privacy "cost" of each of these $O(\frac{\varepsilon}{a} + \frac{\varepsilon}{b})$ elements of a user's transcript coming from a different distribution and bound the sum of the costs by ε. □

4.2 Accuracy Guarantee

Our accuracy theorem needs the following assumption on L, the size of the smallest subgroup, to guarantee that a global update occurs whenever any subgroup has all of its member users vote "yes".

Assumption 4.2. $L > \left(\frac{3}{\sqrt{2}} + \frac{\sqrt{32}}{\varepsilon}\right)\sqrt{n\ln(12mT/\delta)}$.

This brings us to our accuracy theorem, followed by a proof sketch (see Supplement for full details).

Theorem 4.3. *Given number of users n, number of subgroups m, smallest subgroup size L, number of rounds R, privacy parameter ε, and chosen epoch length ℓ and number of epochs $T = R/\ell$, with*

Algorithm 1 Global Algorithm: THRESH

Require: number of users n, number of epochs T, minimum subgroup size L, number of subgroups m, epoch length ℓ, privacy parameter ε, failure parameter δ

Initialize global estimate $\tilde{p}^0 \leftarrow -1$
Initialize vote privacy counters $c_1^V, \ldots, c_n^V \leftarrow 0, \ldots, 0$
Initialize estimate privacy counters $c_1^E, \ldots, c_n^E \leftarrow 0, \ldots, 0$
Initialize vote noise level $a \leftarrow \frac{4\sqrt{2n\ln(12mT/\delta)}}{L - \frac{3}{\sqrt{2}}\sqrt{n\ln(12mT/\delta)}}$
Initialize estimate noise level $b \leftarrow \frac{\sqrt{2\ln(12T/\delta)/2n}}{\log(T)\sqrt{\ln(12nT/\delta)/2\ell} - \sqrt{\ln(12T/\delta)/2n}}$
for each epoch $t \in [T]$ **do**
  for each user $i \in [n]$ **do**
    User i publishes $a_i^t \leftarrow \text{VOTE}(i, t)$
  end for
  GlobalUpdate$^t \leftarrow \left(\frac{1}{n}\sum_{i=1}^n a_i^t > \frac{1}{e^a+1} + \sqrt{\frac{\ln(10T/\delta)}{2n}}\right)$
  if GlobalUpdate^t **then**
    $f(t) \leftarrow t$
    for each $i \in [n]$ **do**
      User i publishes $\tilde{p}_i^t \leftarrow \text{EST}(i, t)$
    end for
    Aggregate user estimates into global estimate: $\tilde{p}^t \leftarrow \frac{1}{n}\sum_{i=1}^n \frac{\tilde{p}_i^t(e^b+1)-1}{e^b-1}$
  else
    $f(t) \leftarrow f(t-1)$
    for each $i \in [n]$ **do**
      User i publishes $\tilde{p}_i^t \leftarrow \mathsf{Ber}(\frac{1}{e^b+1})$
    end for
    $\tilde{p}^t \leftarrow \tilde{p}^{t-1}$
  end if
  Analyst publishes $\tilde{p}^t$
end for

Algorithm 2 Local Subroutine: VOTE

Require: user i, epoch t

Compute local estimate $\hat{p}_i^t \leftarrow \frac{1}{\ell}\sum_{r \in E^t} x_i^r$
$b^* \leftarrow$ highest b such that $|\hat{p}_i^t - \hat{p}_i^{f(t)}| > \tau_b$
VoteYes$_i^t \leftarrow (c_i^V < \varepsilon/4$ and $2^{\lfloor \log T \rfloor - b^*}$ divides $t)$
if VoteYes$_i^t$ **then**
  $c_i^V \leftarrow c_i^V + a$
  $a_i^t \leftarrow \mathsf{Ber}(\frac{e^a}{e^a+1})$
else
  $a_i^t \leftarrow \mathsf{Ber}(\frac{1}{e^a+1})$
end if
Output a_i^t

Algorithm 3 Local Subroutine: EST

Require: user i, epoch t

SendEstimate$_i^t \leftarrow \{c_i^E < \varepsilon/4\}$
if SendEstimate$_i^t$ **then**
  $c_i^E \leftarrow c_i^E + b$
  $\tilde{p}_i^t \leftarrow \mathsf{Ber}(\frac{1+\hat{p}_i^t(e^b-1)}{e^b+1})$
else
  $\tilde{p}_i^t \leftarrow \mathsf{Ber}(\frac{1}{e^b+1})$
end if
Output $\tilde{p}_i^t$

probability at least $1-\delta$*, in every epoch* $t \in [T]$ *such that fewer than*

$$\frac{\varepsilon}{4} \cdot \min\left(\frac{L}{8\sqrt{2n\ln(12mT/\delta)}} - 1, \frac{1}{\sqrt{2}}\left[\log(T)\sqrt{\frac{n}{\ell}} - 1\right]\right)$$

changes have occurred in epochs $1, 2, \ldots, t$*,* THRESH *outputs* $\tilde{p}^t$ *such that*

$$|\tilde{p}^t - p^t| \leq 4(\lfloor \log(T) \rfloor + 2)\sqrt{\frac{\ln(12nT/\delta)}{2\ell}}.$$

Proof Sketch: We begin by proving correctness of the voting process. We show that (1) if every user decides that their subgroup distribution has not changed then a global update does not occur, (2) if every user in some subgroup decides that a change *has* occurred, then a global update occurs, and (3) for each user i the individual user estimates driving these voting decisions are themselves accurate to within $t_\ell = O(\sqrt{\ln(nT/\delta)/\ell})$ of the true $\mu^t_{g(i)}$. Finally, we prove that if every user decides that a change has occurred, then a global update occurs that produces a global estimate $\tilde{p}^t$ that is within t_ℓ of the true p^t.

To reason about how distribution changes across multiple epochs affect THRESH, we use the preceding results to show that the number of global updates never exceeds the number of distribution changes. A more granular guarantee then bounds the number of changes any user detects—and the number of times they vote accordingly—as a function of the number of distribution changes. These results enable us to show that each change increases a user's vote privacy cap c_i^V by at most 2 and estimate privacy cap c_i^E by at most 1.

Finally, recall that THRESH has each user i compare their current local estimate $\hat{p}_i^t$ to their local estimate in the last global update, $\hat{p}_i^{f(t)}$, to decide how to vote, with higher *thresholds* for $|\hat{p}_i^t - \hat{p}_i^{f(t)}|$ increasing the likelihood of a "yes" vote. This implies that if every user in some subgroup computes a local estimate $\hat{p}_i^t$ such that $|\hat{p}_i^t - \hat{p}_i^{f(t)}|$ exceeds the highest threshold, then every user sends a "yes" vote and a global update occurs, bringing with it the global accuracy guarantee proven above. In turn, we conclude that $|\tilde{p}^t - p^t|$ never exceeds the highest threshold, and our accuracy result follows. □

We conclude this section with a few remarks about THRESH. First, while the provided guarantee depends on the number of changes of any size, one can easily modify THRESH to be robust to changes of size $\leq c$, paying and additive c term in the accuracy. Second, the accuracy's dependence on ℓ offers guidance for its selection: roughly, for desired accuracy α, one should set $\ell = 1/\alpha^2$. Finally, in practice one may want to periodically assess how many users have exhausted their privacy budgets, which we can achieve by extending the voting protocol to estimate the fraction of "live" users. We primarily view this as an implementation detail outside of the scope of the exact problem we study.

5 An Application to Heavy Hitters

We now use the methods developed above to obtain similar guarantees for a common problem in local differential privacy known as *heavy hitters*. In this problem each of n users has their own dictionary value $v \in \mathcal{D}$ (e.g. their homepage), and an aggregator wants to learn the most frequently held dictionary values (e.g. the most common homepages), known as "heavy hitters", while satisfying local differential privacy for each user. The heavy hitters problem has attracted significant attention [20, 17, 5, 8]. Here, we show how our techniques combine with an approach of Bassily and Smith [3] to obtain the first guarantees for heavy hitters on evolving data. We note that our focus on this approach is primarily for expositional clarity; our techniques should apply just as well to other variants, which can lead to more efficient algorithms.

5.1 Setting Overview

As in the simpler Bernoulli case, we divide time into $\ell \cdot T$ rounds and T epochs. Here, in each round r each user i draws a sample v_i^r from a subgroup-specific distribution $\mathcal{P}^r_{g(i)}$ over the d values in dictionary $\mathcal{D}$ and track $\mathcal{P}^1, \ldots, \mathcal{P}^T$, the weighted average dictionary distribution in each epoch. We will require the same Assumption 4.2 as in the Bernoulli case, and we also suppose that $d \gg n$, a common parameter regime for this problem.

In the Bernoulli case users could reason about the evolution of μ_j^t directly from their own ℓ samples in each epoch. Since it is reasonable to assume $d \gg \ell$, this is no longer possible in our new setting—$\mathcal{P}_j^t$ is too large an object to estimate from ℓ samples. However, we can instead adopt a common approach in heavy hitters estimation and examine a "smaller" object using a *hash* on dictionary samples. We will therefore have users reason about the distribution p_j^t over hashes that $\mathcal{P}_j^t$ induces, which is a much smaller joint distribution of m (transformed) Bernoulli distributions. Our hope is that users can reliably "detect changes" by analyzing p_j^t, and the feasibility of this method leans crucially on the properties of the hash in question.

5.2 Details and Privacy Guarantee

First we recall the details of the one-shot protocol from Bassily and Smith [3]. In their protocol, each user starts with a dictionary value $v \in [d]$ with an associated basis vector $e_v \in \mathbb{R}^d$. The user hashes this to a smaller vector $h \in \mathbb{R}^w$ using a (population-wide) Φ, a $w \times d$ Johnson-Lindenstrauss matrix where $w \ll d$. The user then passes this hash $\hat{z}_i^t = \Phi e_v$ to their own local randomizer $\mathcal{R}$, and the center aggregates these randomized values into a single $\bar{z}$ which induces a frequency oracle.

We will modify this to produce a protocol HEAVYTHRESH in the vein of THRESH. In each epoch t each user i computes an estimated histogram $\hat{p}_i^t$ and then hashes it into $\Phi\hat{p}_i^t \in \mathbb{R}^w$, where $w = 20n$ (we assume the existence of a subroutine GenProj for generating Φ). Each user votes on whether or not a global update has occurred by comparing $\Phi\hat{p}_i^t$ to their estimate during the most recent update, $\Phi\hat{p}_i^{f(t)}$, in HEAVYVOTE. Next, HEAVYTHRESH aggregates these votes to determine whether or not a global update will occur. Depending on the result, each user then calls their own estimation subroutine HEAVYEST and outputs a randomized response using $\mathcal{R}$ accordingly. If a global update occurs, HEAVYTHRESH aggregates these responses into a new published global hash $\tilde{y}^t$; if not, HEAVYTHRESH publishes $\tilde{y}^{t-1}$. In either case, HEAVYTHRESH publishes $(\Phi, \tilde{y}^t)$ as well. This final output is a *frequency oracle*, which for any $v \in [d]$ offers an estimate $\langle \Phi e_v, \tilde{y}^t \rangle$ of $\mathcal{P}^t(v)$.

HEAVYTHRESH will use the following thresholds with $\tau_b = 2(b+1)\sqrt{2\ln(16wnT/\delta)/w\ell}$ for $b = -1, 0, \ldots, \lfloor \log(T) \rfloor$. See Section 5.3 for details on this choice. Fortunately, the bulk of our analysis uses tools already developed either in Section 4 or Bassily and Smith [3]. Our privacy guarantee is almost immediate: since HEAVYTHRESH shares its voting protocols with THRESH, the only additional analysis needed is for the estimation randomizer $\mathcal{R}$ (see Supplement). Using the privacy of $\mathcal{R}$, privacy for HEAVYTHRESH follows by the same proof as for the Bernoulli case.

Theorem 5.1. HEAVYTHRESH *is ϵ-local differentially private.*

5.3 Accuracy Guarantee

As above, an accuracy guarantee for HEAVYTHRESH unfolds along similar lines as that for THRESH, with additional recourse to results from Bassily and Smith [3]. We again require Assumption 4.2 and also assume $d = 2^{o(n^2/\ell)}$ (a weak assumption made primarily for neatness in Theorem 1.2). Our result and its proof sketch follow, with details and full pseudocode in the Supplement.

Theorem 5.2. *With probability at least $1 - \delta$, in every epoch $t \in [T]$ such that fewer than*

$$\frac{\varepsilon}{4} \cdot \min\left(\frac{L}{8\sqrt{2n\ln(12mT/\delta)}} - 1, \frac{\log(T)\sqrt{\frac{n\ln(320n^2T/\delta)}{10\ell}} - \sqrt{\frac{\ln(16dT/\delta)}{10}} - 2\ln(320nT/\delta)\sqrt{\frac{5}{n}}}{\sqrt{\ln(320nT/\delta)}\left(1 + \frac{20}{\sqrt{n}}\right)} \right)$$

changes have occurred in epochs $1, 2, \ldots, t$,

$$|\hat{f}^t(v) - \mathcal{P}^t(v)| < 4(\log(T) + 2)\sqrt{\frac{2\ln(320n^2T/\delta)}{\ell}} + \sqrt{\frac{\ln(\frac{16ndT}{\delta})}{n}}.$$

Proof Sketch: Our proof is similar to that of Theorem 4.3 and proceeds by proving analogous versions of the same lemmas, with users checking for changes in the subgroup distribution over observed *hashes* rather than observed *bits*. This leads to one new wrinkle in our argument: once we show that the globally estimated hash is close to the true hash, we must translate from closeness of hashes to closeness of the distributions they induce . The rest of the proof, which uses guarantees of user estimate accuracy to 1. guarantee that sufficiently large changes cause global updates and 2. each change incurs a bounded privacy loss, largely follows that of Theorem 4.3. □

References

[1] John M. Abowd. The challenge of scientific reproducibility and privacy protection for statistical agencies. Census Scientific Advisory Committee, 2016.

[2] Differential Privacy Team Apple. Learning with privacy at scale. Technical report, Apple, 2017.

[3] Raef Bassily and Adam Smith. Local, private, efficient protocols for succinct histograms. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 127–135. ACM, 2015.

[4] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Differentially private empirical risk minimization: Efficient algorithms and tight error bounds. *arXiv preprint arXiv:1405.7085*, 2014.

[5] Raef Bassily, Uri Stemmer, and Abhradeep Guha Thakurta. Practical locally private heavy hitters. In *Advances in Neural Information Processing Systems*, pages 2285–2293, 2017.

[6] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Usharsee Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. *arXiv preprint arXiv:1710.00901*, 2017.

[7] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to noninteractive database privacy. *Journal of the ACM (JACM)*, 60(2):12, 2013.

[8] Mark Bun, Jelani Nelson, and Uri Stemmer. Heavy hitters and the structure of local privacy. *arXiv preprint arXiv:1711.04740*, 2017.

[9] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(Mar):1069–1109, 2011.

[10] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Proceedings of Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017.

[11] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

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

[13] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N Rothblum, and Salil Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 381–390. ACM, 2009.

[14] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 715–724. ACM, 2010.

[15] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067. ACM, 2014.

[16] Moritz Hardt and Guy N Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 61–70. IEEE, 2010.

[17] Justin Hsu, Sanjeev Khanna, and Aaron Roth. Distributed private heavy hitters. In *International Colloquium on Automata, Languages, and Programming*, pages 461–472. Springer, 2012.

[18] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? In *Proceedings of the 54th Annual Symposium on Foundations of Computer Science*, pages 531–540, 2008.

[19] Katrina Ligett, Seth Neel, Aaron Roth, Bo Waggoner, and Steven Z Wu. Accuracy first: Selecting a differential privacy level for accuracy constrained erm. In *Advances in Neural Information Processing Systems*, pages 2563–2573, 2017.

[20] Nina Mishra and Mark Sandler. Privacy via pseudorandom sketches. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 143–152. ACM, 2006.

[21] Ryan M Rogers, Aaron Roth, Jonathan Ullman, and Salil Vadhan. Privacy odometers and filters: Pay-as-you-go composition. In *Advances in Neural Information Processing Systems*, pages 1921–1929, 2016.

[22] Aaron Roth and Tim Roughgarden. Interactive privacy via the median mechanism. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 765–774. ACM, 2010.

[23] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and Xiaofeng Wang. Privacy loss in apple's implementation of differential privacy on macos 10.12. *arXiv preprint arXiv:1709.02753*, 2017.

[24] Jonathan Ullman. Tight lower bounds for locally differentially private selection. *Manuscript*, 2018.