Differential Privacy Under Continual Observation

Cynthia Dwork
Microsoft Research
Mountain View, CA, USA
dwork@microsoft.com

Moni Naor*
Weizmann Institute of Science
Rehovot, Israel
moni.naor@weizmann.ac.il

Toniann Pitassi†
University of Toronto
Toronto, Canada
toni@cs.toronto.edu

Guy N. Rothblum‡
Princeton University
Princeton, NJ, USA
rothblum@alum.mit.edu

ABSTRACT

Differential privacy is a recent notion of privacy tailored to privacy-preserving data analysis [11]. Up to this point, research on differentially private data analysis has focused on the setting of a trusted curator holding a large, static, data set; thus every computation is a "one-shot" object: there is no point in computing something twice, since the result will be unchanged, up to any randomness introduced for privacy.

However, many applications of data analysis involve repeated computations, either because the entire goal is one of monitoring, *e.g.*, of traffic conditions, search trends, or incidence of influenza, or because the goal is some kind of adaptive optimization, *e.g.*, placement of data to minimize access costs. In these cases, the algorithm must permit continual observation of the system's state. We therefore initiate a study of *differential privacy under continual observation*. We identify the problem of maintaining a counter in a privacy preserving manner and show its wide applicability to many different problems.

Categories and Subject Descriptors

F.2.0 [**Theory of Computation**]: Analysis of Algorithms and problem complexity—*General*

General Terms

Algorithms, Security, Theory, Privacy, Private Data Analysis

1. INTRODUCTION

Differential privacy is a recent privacy guarantee tailored to the problem of statistical disclosure control: how to publicly release statistical information about a set of people without compromising the privacy of any individual [11] (see [8, 9, 6] for motivation of the definition, history, basic techniques, and pointers to recent results). In a nutshell, differential privacy requires that the probability distribution on the published results of an analysis is "essentially the same," independent of whether any individual opts in to, or opts out of, the data set. (The probabilities are over the coin flips of the privacy mechanism.) Statistical databases are frequently created to achieve a social goal, and increased participation in the databases permits more accurate analyses. The differential privacy guarantee supports the social goal by assuring each individual that she incurs little risk by joining the database: anything that can happen is essentially equally likely to do so whether she joins or abstains.

Up to this point, research on differentially private data analysis has focused on the setting of a trusted curator holding a large, static, data set, in a permanently infrangible storage system. The curator either responds to queries (the *interactive* case) or prepares some sort of summary or synthetic database intended to answer all queries of a particular type (the *non-interactive* case)[1]. A line of work investigates the way in which the accuracy of the responses to the queries may need to deteriorate with the number, sensitivity (a metric describing how much the addition or deletion of a member of the database can change affect the outputs), and geometry of the query sequence [5, 12, 16, 17], and general techniques have been developed that in some cases match these bounds [15, 2, 11, 19, 17, 3, 14].

In all previous work every computation is a "one-shot" object: there is no point in computing something twice, since the result will be unchanged, up to any perturbations introduced for privacy. However, many applications of data analysis involve repeated computations, either because the

*Research supported by a grant from the Israel Science Foundation. Part of this work was done while visiting MSR and the Center for Computational Intractability at Princeton University.

†Research supported by Nserc. Part of this work was done while visiting MSR.

‡Research supported by NSF Grants CCF-0635297, CCF-0832797 and by a Computing Innovation Fellowship. Parts of this work were done while visiting and interning at MSR.

STOC'10, June 5–8, 2010, Cambridge, Massachusetts, USA.

[1]The one exception considers a distributed setting, in which each data holder controls her own data and decides in which analyses to participate [10]. However, this is done by emulating the curator via secure multiparty computation.

entire goal is one of monitoring, *e.g.*, of traffic conditions, search trends, or incidence of influenza, typically with a goal of preparing some appropriate response, or the goal is to optimize the placement of data to minimize access costs. In such an application the system is required to continually produce outputs. We therefore initiate a study of *differential privacy under continual observation*. We illustrate the design goals of our continual observation algorithms by appeal to a real-life example.

Continual Observation. Consider a website for H1N1 self-assessment[2]. Individuals can interact with the site to learn whether symptoms they are experiencing may be indicative of the H1N1 flu. The user fills in some demographic data (age, zipcode, sex), and responds to queries about his symptoms (fever over 100.4°F?, sore throat?, duration of symptoms?). We would like to *continually* analyze aggregate information of consenting users in order to monitor regional health conditions, with the goal, for example, of organizing improved flu response. Can we do this in a differentially private fashion with reasonable accuracy (despite the fact that the system is continually producing outputs)?

We would expect a given individual to interact very few times with the H1N1 self-assessment site. For simplicity, let us say this is just once (the general case is an easy extension). In such a setting, it is sufficient to ensure *event-level* privacy, in which the privacy goal is to hide the presence or absence of a single event (interaction of one user with the self-assessment site). That is, the probability of any output sequence should be essentially the same, independent of the presence or absence of any single interaction with the site.

The privacy statement at the H1N1 self-assessment site encourages users to allow their data to be shared:

> "This information can be very helpful in monitoring regional health conditions, plan flu response, and conduct health research. By allowing the responses to the survey questions to be used for public health, education and research purposes, you can help your community."

But the site also describes conditions under which the data may be disclosed:

> "...if required to do so by law or in the good faith belief that such action is necessary to (a) conform to the edicts of the law or comply with legal process served on Microsoft or the Site; (b) protect and defend the rights or property of Microsoft and our family of Web sites; or (c) act in urgent circumstances to protect the personal safety of users of Microsoft products or members of the public."

Pan-Privacy. This raises the following question, orthogonal to that of continual observation: Is it possible to maintain a differentially private *internal state*, so that a consenting user has privacy even against a subpoena or other intrusion into the local state? Algorithms with this property are called *pan-private*, as they are private inside (internal state) and out (output sequence). This notion was recently introduced in [13]. The goal of pan-privacy is to provide a mathematically rigorous way of (essentially) eliminating the risk – of anything – incurred by sharing one's information, *even in the presence of an intrusion*, further encouraging participation. Another motivation is the prevention of "mission creep" for data, protecting the data curator from the (very real!) pressure to allow data to be used for purposes other than that for which they were collected. This fits well with streaming algorithms with small state, which can at most store a few data items, and [13] does focus on the streaming model. However, nothing prevents a streaming algorithm from storing in its state an event of a person of interest. Event-level pan-privacy rules this out, since the internal state must have essentially the same distribution, independent of whether an event of the person of interest has, or has not, appeared in the stream.

We remark that, since an intrusion can occur at an unpredictable time, designing pan-private algorithms is interesting even when the system will generate only a single output. This was studied in [13]. The current paper addresses pan-privacy under continual observation. We emphasize that continual output and pan-privacy are each interesting and non-trivial properties and, generally speaking, they are independent of each other[3]. Their combination is, of course, more powerful than each individually.

Adjacency notions: User-Level vs. Event-Level Privacy. We have made the reasonable assumption that a single individual interacts with the H1N1 self-assessment site at most a small number of times (one). Such an assumption is not reasonable for other kinds of websites, such as a search engine. We ask: what kinds of statistics can we gather while preserving the privacy of an individual's entire history of accesses to the website? That is, what can we compute if we require that, no matter how many times an individual accesses the website, and no matter how these accesses are interleaved with those of others, the distribution on outputs, or, in the case of *user-level pan-privacy*, the distribution on pairs (internal state, output sequence), should be essentially the same, independent of presence or absence of any individual's data in the stream? All the algorithms of [13] enjoy user-level pan-privacy. In this paper we construct several algorithms that exhibit user-level pan-privacy under continual observation.

2. OVERVIEW OF OUR RESULTS

Counter. We identify the problem of maintaining a counter in a privacy preserving manner and show its wide applicability to many different problems. We construct a continual observation low-error pan-private *counter* estimating, at all times, the answer to the question "How many times has an event of interest happened so far?"

One-shot → Continual Observation. We describe a general transformation that, roughly speaking, converts a large class of (possibly user-level possibly pan-private) one-shot algorithm and produces an algorithm with the same types of privacy guarantees under continual observation. This class contains algorithms for monotonically increasing or decreasing functions with a bounded range, as well as functions that are "close to" monotonic, provided the algorithm satisfies a condition we call (k, d)-unvarying. Roughly speaking, this condition says that the output can change by an additive d

[2]https://h1n1.cloudapp.net is such a website. User-supplied data are stored for analysis only if the user consents.

[3]The extreme case of continual intrusion also implies continual output

amount at most k times. The loss of accuracy deteriorates as k and d increase.

Applications of the Counter. Counting is a fundamental computational primitive. We illustrate the utility of the counter by obtaining event-level pan-private algorithms for sophisticated tasks, such as following expert advice. We obtain an event-level pan-private under continual observation version of *Follow the Perturbed Leader* [18].

Negative Results. Pan-privacy comes at a price that increases with the number of intrusions: we prove that any counter providing pan-privacy against a number k of intrusions on the internal state must have error at least $\sqrt{k}$. In [13] Dwork *et al.* obtained an accurate, one-shot, pan-private algorithm for modular incidence counting. Unlike all the other problems addressed in that work, for this problem there is no reasonably accurate algorithm providing user-level privacy under continual observation, even without pan-privacy. This result generalizes to any function whose value "many times" increases by "a lot" (the accuracy deteriorates with the product of "many" and "a lot;" see Section 4.2).

3. DEFINITIONS AND TOOLS

3.1 Differential Privacy Basics

In the literature, a differentially private mechanism operates on a *database*, or *data set*. This is a collection of *rows*, where the data of an individual are held in a single row. Differential privacy ensures that the ability of an adversary to inflict harm (or good, for that matter) – of any sort, to any set of people – is essentially the same, independent of whether any individual opts in to, or opts out of, the dataset. This is done indirectly, simultaneously addressing all possible forms of harm and good, by focusing on the probability of any given output of a privacy mechanism and how this probability can change with the addition or deletion of any row. We will concentrate on pairs of databases (D, D') differing only in one row, meaning one is a subset of the other and the larger database contains just one additional row.

DEFINITION 3.1. *[11] A randomized function $\mathcal{K}$ gives* ε-differential privacy *if for all data sets D and D' differing in at most one row, and all $S \subseteq Range(\mathcal{K})$,*

$$\Pr[\mathcal{K}(D) \in S] \quad \leq \quad \exp(\varepsilon) \times \Pr[\mathcal{K}(D') \in S], \qquad (1)$$

where the probability is over the coin flips of $\mathcal{K}$.

The multiplicative nature of the guarantee implies that an output whose probability is zero on a given database must also have probability zero on any neighboring database, and hence, by repeated application of the definition, on any other database. The parameter ε is public, and its selection is a social question. We tend to think of ε as, say, 0.01, 0.1, or in some cases, $\ln 2$ or $\ln 3$.

DEFINITION 3.2. *[11] For $f : \mathcal{D} \to \mathbf{R}^d$, the* L_1 sensitivity *of f is $\Delta f = \max_{D,D'} \|f(D) - f(D')\|_1$ for all D, D' differing in at most one row.*

The Laplace distribution with parameter b, denoted $Lap(b)$, has density function $\Pr(z|b) = \frac{1}{2b}\exp(-|z|/b)$ and variance $2b^2$. Taking $b = 1/\varepsilon$ we have that the density at z is proportional to $e^{-\varepsilon|z|}$. This distribution has highest density at 0 (good for accuracy), and for any z, z' such that $|z - z'| \leq 1$ the density at z is at most e^{ε} times the density at z'. Finally, the distribution gets flatter as ε decreases: smaller ε means better privacy, so the noise density should be less "peaked" at 0 and change more gradually as the magnitude of the noise increases.

THEOREM 3.1. *[11] For $f : \mathcal{D} \to \mathbf{R}^d$, the mechanism $\mathcal{K}$ that adds independently generated noise with distribution $Lap(\Delta f/\varepsilon)$ to each of the d output terms enjoys ε-differential privacy.*

THEOREM 3.2. *[11] The composition of an ε_1-differentially private mechanism and an ε_2-differentially private mechanism is at worst $(\varepsilon_1 + \varepsilon_2)$-differentially private.*

3.2 Event-Level and User-Level (Pan-)Privacy

Speaking intuitively, we think of continual observation algorithms as taking steps at discrete time intervals; at each step the algorithm receives an input, computes, and produces output. We model this formally with a streaming algorithm. Computation proceeds in a sequence of atomic steps. At each step the algorithm receives an input from the stream, computes (changes state), and produces outputs. Thus the intuitive notion of "t time periods" corresponds to processing a sequence of t elements in the stream. Intrusions may only occur *between* atomic steps.

Because we are modeling real systems, where time is a factor (computers have clocks, as do the adversaries), we will occasionally need to model the fact that "nothing has happened" in a given time unit. We may do this by means of a "nothing happened" element. For example, the motivation behind the counter primitive below is to count the number of times that something has occurred since the algorithm was started (the counter is very general; we don't specify *a priori* what it is counting). This is modeled by an input stream over $\{0, 1\}$. Here, "0" means "nothing happened," "1" means the event of interest occurred, and for $t = 1, 2, \ldots$ the algorithm outputs an approximation to the number of 1's seen in the length t prefix of the stream.

To define privacy, we need a notion of adjacent stream prefixes. The definition of adjacency in [13] permitted strings of radically different lengths to be adjacent. To be able to capture time, we modify the definition as follows. Let X be the universe of possible input symbols. Let S and S' be stream prefixes (ie, finite streams) of symbols drawn from X. Then $\mathrm{Adj}(S, S')$ ("S is adjacent to S'") if and only if there exist $x, x' \in X$ so that if we change some of the instances of x in S to instances of x', then we get S'. More formally, $\mathrm{Adj}(S, S')$ iff $\exists x, x' \in X$ and $\exists T \subseteq [|S|]$, such that $S|_{T:x\to x'} = S'$. Here, T is a set of indices in the stream prefix S, and $S|_{T:x\to x'}$ is the result of replacing all the occurrences of x at these indices with x' (note that w.l.o.g we can assume $S|_T$ contains only occurrences of x).

The definition of adjacency anticipates user-level privacy, where the behavior of the algorithm should remain the same independent of the number or pattern of instances of any single symbol x in the stream. A special case of adjacency is *event-level adjacency*, in which the number of instances of one symbol replaced by another is at most 1. Naturally, this will be used for defining event-level privacy. With these definitions, adjacent prefixes are always of the same length.

DEFINITION 3.3. *Let* **Alg** *be an algorithm. Let* I *denote the set of internal states of the algorithm, and σ the set of possible output sequences. Then algorithm* **Alg** *mapping data*

stream prefixes to the range $\mathrm{I} \times \sigma$*, is* user-level pan-private (against a single intrusion) *if for all sets* $\mathrm{I}' \subseteq \mathrm{I}$ *and* $\sigma' \subseteq \sigma$*, and for all pairs of* X*-adjacent data stream prefixes* S, S'

$$\Pr[\mathbf{Alg}(S) \in (\mathrm{I}', \sigma')] \leq e^{\varepsilon} \Pr[\mathbf{Alg}(S') \in (\mathrm{I}', \sigma')]$$

where the probability is over the coin flips of **Alg**.

We typically omit the term ε-differential when discussing pan-privacy. However, every claim of pan-privacy must describe the privacy loss.

Event-level pan-privacy is defined analogously, with the requirement restricted to event-level adjacent streams. In the case of one-shot algorithms the output sequence has length 1. In the case of continual observation the output sequence is as long as the stream prefix processed by the algorithm. Ordinary user-level, respectively, event-level, privacy (as opposed to pan-privacy) is just differential privacy for user-level, respectively, event-level, adjacent streams.

Note that popular techniques from the streaming literature, such as Count-Min Sketch and subsampling, cannot be pan-private.

4. EVENT-LEVEL PRIVATE COUNTING

In this section we introduce the web-counter problem, and present algorithms and lower bounds.

Monitoring events, such as the spread of an epidemic or the traffic at a website, frequently can be cast in terms of counting the number of "interesting" events that have occurred over time. Formally, we view time as a series of time periods, named by the natural numbers. In each time period there are two possibilities: an event occurs or no event occurs. We model this as a binary stream, with 1's corresponding to periods in which an event occurs and 0's corresponding to periods in which no event occurs. Informally, the *web counter problem* is to continually produce an accurate, but not necessarily exact, estimate of the number of events that have occurred "so far," while obscuring, for each time period t, whether or not an event actually occurs at t.

Definition 4.1. *A randomized streaming algorithm yields a* (T, α, β) counter *if, in every execution, with probability at least* $1 - \beta$ *over the coin flips of the algorithm, simultaneously, for all* $1 \leq t \leq T$*, after processing a prefix of length* t*, the current output contains an estimate of the number of 1's in the prefix that differs from the true weight of the prefix by at most an additive* α *amount.*

with T and $1/\beta$. In the remainder of this section we present an event-level counter[4] for which α grows as roughly $\log(1/\beta)\log^{2.5} T$. There is a narrow gap between this and the lower bound (see Section 4.1). The algorithm is pan-private against a single intrusion; it appears in Figure 1.

It will be convenient to number the time periods $0, 1, \ldots, T-1$, so that in time period t the algorithm is processing the $(t+1)$st element in the input stream. Similarly, we will number the elements of the stream beginning with 0, so on stream x at time t the algorithm is processing x_t. The algorithm requires the following set of (time) *segments*. For $i \in \{1, \ldots, \log T\}$, associate with each string $s \in \{0,1\}^i$ the time segment S of $2^{\log T - i}$ time periods $\{s \circ 0^{\log T - i}, \ldots s \circ 1^{\log T - i}\}$. We say the segment *begins in time* $s \circ 0^{\log T - i}$ and *ends in time* $s \circ 1^{\log T - i}$.

Theorem 4.1. *The counter algorithm of Figure 1 run with parameters* $T, \varepsilon, \beta > 0$*, yields a* $(T, 4\log(1/\beta) \cdot \log^{2.5} T/\varepsilon, \beta)$ *counter enjoying* ε*-differential pan-privacy against a single intrusion.*

Proof. We first argue about the accuracy of the algorithm, and then prove event-level differential pan-privacy.

Accuracy. Let S be a segment. The noise value η_S is stored in the algorithm's internal state during the first time period contained in S, and is deleted during the atomic step occurring in the last time period contained in S. Thus, all the noise values for segments containing t are in memory in Step 3 when we generate time period t's output.

We add $1 + \log T$ independently chosen Laplace noise variables to the true answer: the $\log T$ segment noises, and the noise with which we initialize *count*. Each of these variables has variance $2(\log T/\varepsilon)^2$, Thus, in each round, with probability at least $1 - \beta/T$ probability, the sum of noises is within magnitude $2\log(1/\beta) + \log T$ standard deviations, so by a union bound, with probability at least $1 - \beta$, in all rounds simultaneously, this sum of noises has magnitude smaller than $4\log(1/\beta) \cdot \log^{2.5} T/\varepsilon$.

Pan-Privacy. During an intrusion between atomic steps t^* and $t^* + 1$, that is, immediately following the processing of element t^* in the input stream (recall that we begin numbering the elements with 0), the view of the adversary consists of (1) the noisy count, (2) the segment noise values η_S in memory when the intrusion occurs, and (3) the complete sequence of all of the algorithm's outputs. Consider adjacent databases x and x', which differ in time t, say, without loss of generality, $x_t = 1$ and $x'_t = 0$, and an intrusion immediately following time period $t^* \geq t$ (we will discuss the case $t^* < t$ below). We will describe a bijection between the vector of noise values used in executions on x and executions on x', such that corresponding noise values induce identical adversary views on x and x', and the probabilities of adjacent noise values differ only by an e^{ε} multiplicative factor. This implies ε-differential pan-privacy.

By assumption, the true count just after the time period $t^* \geq t$ is larger when the input is x than it is when the input is x'. Fix an arbitrary execution E_x when the input stream is x. This amounts to fixing the randomness of the algorithm, which in turn fixes the noise values generated. We will describe the corresponding execution $E_{x'}$ by describing how its noise values differ from those in E_x.

The program variable *count* was initialized with Laplace noise. By increasing this noise by 1 in $E_{x'}$ the value of *count* just after step t^* is identical in $E_{x'}$ and E_x. The noise variables in memory immediately following period t^* are independent of the input; these will be unchanged in $E_{x'}$. We will make the sequence of outputs in $E_{x'}$ *identical* to those in E_x by changing a collection of $\log T$ segment noise values η_S that are *not* in memory when the adversary intrudes, so that the sum of *all* noise values in all rounds up through $t-1$ is unchanged, but the sum from round t on is larger by 1 for database x' than for x. Since we *increased* the initialization noise for *count*, we now need to *decrease*

[4]Our original algorithm (see full version of this paper or [7]) had worse performance guarantees. Several people, including Ilya Mironov, Kobbi Nissim, and Adam Smith (thanks guys!), immediately suggested an approach closely resembling the one presented here, based on the Bentley-Saxe method of converting static data structures to dynamic ones [1]. Our algorithm is slightly different, in order to ensure pan-privacy.

Counter (T, ε)

Initialization. Initialize $\xi = (1 + \log T)/\varepsilon$, and sample $count \sim Lap(\xi)$.

Segments. For $i \in \{1, \ldots, \log T\}$, associate with each string $s \in \{0,1\}^i$ the time segment S of $2^{\log T - i}$ time periods $\{s \circ 0^{\log T - i}, \ldots s \circ 1^{\log T - i}\}$. The segment *begins in time* $s \circ 0^{\log T - i}$ and *ends in time* $s \circ 1^{\log T - i}$.

Processing. In time period $t \in \{0, 1, \ldots, T-1\}$, let $x_t \in \{0,1\}$ be the t-th input bit:

1. $count \leftarrow count + x_t$;
2. For every segment S which begins in time t, sample noise $\eta_S \sim Lap(\xi)$;
3. Let $S_1, \ldots, S_{\log T}$ be the $\log T$ segments that contain t. Output $count + \sum_{i=1}^{\log T} \eta_{S_i}$.
4. For every segment S that ends in time t, erase η_S.

Figure 1: Event-Level Counter Algorithm

the sum of segment noise values for periods $0, \ldots, t-1$ by 1, and leave unchanged the sum of segment noise values from period t.

To do this, we find a collection of disjoint segments whose union is $\{0, \ldots, t-1\}$. There is always such a collection, and it is always of size at most $\log T$. We can construct it iteratively by, for i decreasing from $\lfloor \log(t-1) \rfloor$ to 0, choosing the segment of size 2^i that is contained in $\{0, \ldots, t-1\}$ and is not contained in a previously chosen segment (if such a segment exists). Given this set of disjoint segments, we notice also that they all end at time $t - 1 < t \leq t^*$, and so their noises are not in memory when the adversary intrudes (just following period t^*). In total (taking into account also changing the initial noise value for *count*), the complete view seen by the adversary is identical and the probabilities of the (collection of) noise values used for x and x' differ by at most an e^{ε} multiplicative factor.

Note that we assumed $t^* \geq t$. If $t^* < t$ then the initial noise added to *count* in $E_{x'}$ will be the same as in E_x, and we need to add 1 to the sum of segment noises in every time period from t through T (the sum of segment noises before time t remains unchanged). This is done as above, by finding a disjoint collection of at most $\log T$ segments that exactly covers $\{t, \ldots, T-1\}$. The noise values for these segments are not yet in memory when the intrusion occurs in time $t^* < t$, and the proof follows similarly. □

We note also that in every round *individually* (rather than in all rounds simultaneously), with all but β probability, the error has magnitude at most $O(\log(1/\beta) \cdot \log^{1.5} T/\varepsilon)$.

4.1 A Logarithmic (in T) Lower Bound

Given the upper bound of Theorem 4.1, where the error depends only poly-logarithmically on T, it is natural to ask whether *any* dependence is inherent. In this section we show that a logarithmic dependence on T is indeed inherent. The proof is omitted from this extended abstract.

THEOREM 4.2. *Any differentially private event-level algorithm for counting over T rounds must have error $\Omega(\log T)$ (even with $\varepsilon = 1$).*

4.2 Lower Bound for Counting with Intrusions

THEOREM 4.3. *Any counter that is ε-pan-private under continual observation while tolerating n intrusions must have additive error $\Omega(\sqrt{n})$ with probability at least $1/2 - 2\delta$, for any $\delta > 0$.*

PROOF. Let A be an ε-pan-private algorithm for counting the number of 1's in x, $|x| = n$, with continual intrusions. At each step, A reads the next bit, x_i, of x, updates its state from s_{i-1} to s_i, and must reveal the entire state s_i (including the updated count.)

Fix a constant k that depends on δ. We define two distributions on inputs of length n: I_0 and I_1. I_0 puts all of its weight on the all 0 string. A string is drawn from I_1 as follows. For each bit position i, set $i = 0$ with probability $1 - 1/k\sqrt{n}$ and set $i = 1$ with probability $1/k\sqrt{n}$. We will show that for any continual-intrusion pan-private algorithm, A, the statistical distance between D_0 and D_1 is at most δ, where D_0 is the distribution on states when running A on I_0, and D_1 is the distribution on states when running A on I_1. Thus there exists two input sequences, s_0, the all 0 input, and s^*, an input with $O(\sqrt{n})$ 1's and the rest zeroes, such that the statistical distance between the distribution of states when run on s_0 versus s^* is at most δ. This in turn implies that for one of these two inputs, the algorithm must make an additive $\Omega(\sqrt{n})$ error with probability at least $1/2 - 2\delta$. (If on input s^* A outputs a count that is smaller than $k'\sqrt{n}$ with probability at least $1/2$, then we are done. Otherwise A outputs a count that is at least $k'\sqrt{n}$ with probability at least $1/2$ on s^*. In this case, since the statistical distance between the states on s_0 and s^* is at most δ, this implies that with probability at least $(1/2 - 2\delta)$, A outputs a count that is at least $k'\sqrt{n}$ on s_0, completing the argument.)

It is left to show that for any ε-pan-private algorithm A with continuous intrusion, the statistical distance between D_0 and D_1 is at most δ. Let $p_0^s(x)$ be the probability, over the random coin tosses of algorithm A, of going to state x from state s, upon seeing a 0. Similarly let $p_1^s(x)$ be the probability of going to state x from state s, upon seeing a 1. We want to compare two distributions: the distribution D_0 induced on the possible states of A, when running A starting in s_0 on the distribution I_0 for n steps, and the distribution D_1 induced on the states of A when running A on I_1 for n steps, still starting from s_0. Let $q_0^s = p_0^s$ and let $q_1^s = (1 - \frac{1}{k\sqrt{n}})p_0^s + \frac{1}{k\sqrt{n}}p_1^s$. D_0 is the distribution induced on the states if you start in the start state, s_0, and run for n steps using probabilities q_0^s and similarly D_1 is the distribution induced on the states running for n steps, using the probabilities q_1^s.

The details of the proof are omitted for lack of space. We provide some intuition and the statements of the key lemma. *Randomized response* is a popular technique in the social sciences for obtaining answers to questions about embarrassing

or even illegal behaviors [20]. We can abstract randomized response as follows. Consider a question for which there are two possible answers, say, $\{0,1\}$. We define two trivial distributions: $Zero$ and One, which place all their weight, respectively, on 0 and 1. We now define two *answer* distributions A_0 and A_1. A respondent whose answer is $b \in \{0,1\}$ chooses from A_b. Both A_0 and A_1 are convex combinations of $Zero$ and one. The intuition for randomized response is that the probability of producing any specific value $v \in \{0,1\}$ when drawing from A_0 is sufficiently close to the probability of producing v when drawing from A_1 to provide "plausible deniability" to the respondent, while the statistician, knowing the descriptions of A_0 and A_1 can "reverse engineer" the responses to obtain statistical information about the distribution of true answers. This completes the abstract description of randomized response. The intuition for our proof is that any counter that provides privacy is carrying out some form of randomized response.

Formally, we show (Lemma 4.4) the existence of two specific distributions $C'(x)$ and $C''(x)$ such that the distributions $q_0^s(x)$ and $q_1^s(x)$ are specific convex combinations of these distributions, say, $q_0^s(x) = z_0C'(x) + z_1C''(x)$ and $q_1^s(x) = w_0C'(x) + w_1C''(x)$. To upper bound the distance between $q_0^s(x)$ and $q_1^s(x)$ we replace $C'(x)$ and $C''(x)$ with distributions that are, intuitively, as far apart as possible, specifically, we replace $C'(x)$ with the trivial $Zero$ distribution and $C''(x)$ with the trivial One distribution, and bound the distance between $z_0Zero+z_1One$ and $w_0Zero+w_1One$.

LEMMA 4.4. *If the algorithm is ε-differentially pan private, then for every state s, distributions q_0^s and q_1^s can be written in the following form:*

$$\begin{aligned} q_0^s(x) &= \frac{1}{2}C'(x) + \frac{1}{2}C''(x) \\ q_1^s(x) &= (\frac{1}{2} - \frac{1}{k\sqrt{n}})C'(x) + (\frac{1}{2} + \frac{1}{k\sqrt{n}})C''(x), \text{ where} \\ C'(x) &= \frac{k\sqrt{n}}{2}q_0^s(x) + q_0^s(x) - \frac{k\sqrt{n}}{2}q_1^s(x) \\ C''(x) &= q_0^s(x) - \frac{k\sqrt{n}}{2}q_0^s(x) + \frac{k\sqrt{n}}{2}q_1^s(x) \end{aligned}$$

That is, q_0^s is obtained by choosing either C' or C'' with equal probability, and then sampling uniformly from the chosen distribution; q_1^s is obtained via biased sampling from the distributions C' and C'': choose C'' with probability $(1/2 + 1/k\sqrt{n})$ and C' with probability $(1/2 - 1/k\sqrt{n})$, and then sample uniformly from the chosen distribution.

□

5. A GENERAL TRANSFORMATION

Our main result in this section will be a general transformation that converts a single-output streaming algorithm into a streaming algorithm that continually produces outputs. As an example, consider a single-output algorithm **Alg** for a monotone function, such as counting. Given a stream σ, the true count is the number of 1's in σ, and $\mathbf{Alg}(\sigma)$ produces an approximation to this true count. The transformed algorithm will be a counting algorithm as defined in the previous section of this paper: after processing any prefix of σ the output will be an approximation to the number of 1's in the prefix.

Our transformation works for single algorithms for any monotonic or "nearly monotonic" (see below) functions. User-level privacy and pan-privacy are preserved by the transformation, so if the original algorithm enjoys either or both of these properties then so will the transformed algorithm.

REMARK 5.1. *Throughout this section, we focus on the properties of the* algorithm *being transformed, such as the probability with which it produces output of specific accuracy, and we obtain results describing how these properties change (typically, deteriorate) in the transformed algorithm.*

Fix a single-output streaming algorithm **Alg**. We begin with the notion of a k-snapshot. Intuitively, this is a snapshot of the algorithm's outputs from k different time periods. Of course, our algorithm to be transformed is a single-output algorithm, so there is only one output, and not k outputs. However, for a fixed length T input stream σ and a fixed sequence τ of coin tosses, given a k-tuple $t_1, \ldots, t_k$ of times in $[T]$, we can fix the algorithm's coin tosses and run it on the k prefixes of length, respectively, $t_1, t_2, \ldots, t_k$. Each time we start from the beginning of σ and τ, and obtain a single output. This gives a well defined notion of the outputs at k different times.

DEFINITION 5.1 (k-SNAPSHOT). *For a fixed T-round execution of an algorithm* **Alg** *(i.e. fixing the algorithm's random coins τ and its input stream σ), a* k-snapshot *is a set of k pairs $(t_1, a_1), \ldots, (t_k, a_k)$, where, for $1 \le i \le t$, a_i is the output produced by* **Alg** *when run on the length t_i prefix of σ, using coin sequence τ (not all coins in τ need be read). We say that a snapshot is* d-varying *if for all $1 \le i \le k$ it holds that $|a_{i+1} - a_i| \ge d$.*

The deterioration in accuracy of the transformed algorithm is affected by the number of times the output can change by at least an additive factor of d. When k and d are both small the accuracy hit will be low.

DEFINITION 5.2 ((k, d, γ)-UNVARYING ALGORITHM). *An algorithm* **Alg** *is (k, d, γ)-unvarying if in every execution, with all but γ probability over the algorithm's coins, there does not exist a d-varying k-snapshot for the execution. I.e., the number of times the algorithm's output varies by at least d is at most k.*

Intuitively, any accurate algorithm computing a monotone function cannot vary too often or too wildly. In particular, when the range of the function to be computed is bounded, for example, when the range is contained in the interval $[0, 1]$, then the output can change by an additive d amount at most $1/d$ times. We now make this precise.

CLAIM 5.1. *Let* **Alg** *be an algorithm for computing a* monotonically increasing *function whose output is in* $[0, 1]$.[5] *If in every execution, with all but γ probability, the outputs of* **Alg** *are all within an α additive error, then for any $0 < d < 1$,* **Alg** *is $(1/d, d + 2\alpha, \gamma)$-unvarying. The claim also holds for monotonically decreasing functions.*

[5] A monotonically increasing function is one whose value can only increase as more data items are processed. For example, density estimation is a monotone increasing function.

PROOF. With probability at least $1-\beta$, the algorithm's outputs are all α-accurate. Assuming we are in this high probability case, for any $k > 0$ consider any k-snapshot $(t_1, a_1), \ldots, (t_k, a_k)$. For any $0 < d < 1$ and any $j \in \{1, \ldots, k\}$, a $d + 2\alpha$-change in the algorithm's output between t_j and t_{j+1} corresponds to a change of at least $1/k$ in the *value of the function* between t_j and t_{j+1} (because of α-accuracy in time periods t_j and in t_{j+1}). Since the function is monotone, this change in its value can only be an increase, and since the output is bounded to be between 0 and 1, there can only be $1/d$ such increases. We conclude that $k \leq 1/d$. □

Before presenting our general transformation, we provide some intuition. Assume **Alg** is a (user-level pan-private) algorithm for a monotonic function f with range in $[0, 1]$. Fix an input stream σ and sequence τ of random coins. Let $\mathbf{Alg}_t$ denote the output that the algorithm would produce after a prefix of length t. Recall that the transformed algorithm will have output at every step; let out_t denote this output after processing the t-length prefix of σ. The first idea is to change *out* infrequently, that is, only when it becomes very inconsistent with $\mathbf{Alg}_t$. In other words, for "suitably chosen" d, at each time t:

> If $\mathbf{Alg}_t - out_{t-1} > d$ then $out_t \leftarrow \mathbf{Alg}_t$,
> else $out_t \leftarrow out_{t-1}$.

As noted above, we will expect at most about $1/d$ changes to *out*. For privacy, we now have two problems. First, if there is no change at time t but there is a change at time $t+1$, this leaks information about σ_{t+1}, the $(t+1)$st symbol in σ, so, still speaking intuitively, we will need to add some noise to the comparison. For a similar reason, we need to add noise to the output.

Since we expect a change to occur only about $1/d$ times, we can scale the noise to this amount and abort the execution if the number of updates exceeds this.

> At time t: if $\mathbf{Alg}_t$ + fresh noise $- out_{t-1} \leq d$ then $out_t \leftarrow out_{t-1}$ (no change in output); else $out_t \leftarrow \mathbf{Alg}_t$ + fresh noise (update output).

This approach leads to an algorithm ensuring a weaker form of privacy known as (ε, δ)-differential privacy. Intuitively, the obstacle to pure differential privacy is that information is also leaked by the fact that an update to *out* does *not* occur. A long sequence of non-updates may be much more likely on an input σ', than on an adjacent input σ. For example, still speaking intuitively, this might happen if the true value of the function on a given prefix is close to triggering an update in σ and slightly farther in σ', and then for a large number of steps there is no input, so the value of the function does not change. Although in each of these steps the ratio of the probability of no update in σ' to the probability of no update in σ may be relatively small, say, e^{ε}, the ratio of probabilities of no update during this entire input-free stretch in σ' to the probability of no update during this stretch in σ may be very large ($e^{m\varepsilon}$ for a stretch of length m). To address this, we introduce a "soft" threshold: instead of comparing to d we compare to d + fresh noise, where fresh noise is chosen anew at each step. This fresh noise will be generated using the same segment-based techniques as were used in the counter construction. These involve relatively few variables at any one time ($\log T$), while simultaneously providing a dynamic set of thresholds for comparison, changing at every step. The general transformation is presented in Figure 2. As we will see, the transformation preserves user-level pan-privacy.

THEOREM 5.2. *For any (k, d, γ)-unvarying algorithm* **Alg** *with (α_0, β_0)-accuracy, ε_0 (pan-)privacy, and user-level sensitivity ρ, the transformation of Figure 2, run with parameters as above and with any $T, \varepsilon, \beta > 0$, yields a T-round continual output algorithm with $(\varepsilon_0 + 3\varepsilon)$ (pan-)privacy. With all but $(\beta + \beta_0 + \gamma)$ probability, the error in all time periods is at most:*

$$d + \alpha_0 + (4\rho \cdot (\log(1/\beta) + \log T) \cdot (k + \log^2 T)/\varepsilon)$$

PROOF. We first argue about the accuracy of the algorithm, and then prove differential pan-privacy (for *pan*-privacy we assume that **Alg** itself is pan-private, otherwise we get only standard differential privacy for continual observation).

Accuracy. Assume that (1) the answers produced by **Alg** are all α-accurate ("**Alg** is accurate"); (2) the sequence of answers returned is (k, d)-unvarying, and (3) all the Laplace noise values have magnitude at most $(\log(1/\beta) + \log T + O(1))$ times their scale ("the noise is never large"). Taking a union bound, all of these conditions above are satisfied (in all rounds simultaneously) with all but $(\beta + \beta_0 + \gamma)$ probability.

If $\widehat{compare}_t \leq \widehat{thresh}_t$, so the conditional in Step 3 of round t evaluates to "true", then we say round t is *light*. Otherwise, that is, if $\widehat{compare}_t > \widehat{thresh}_t$, then we say round t is *heavy*. Fix a round t, and let $h < t$ be the last heavy round preceding round t, if one exists; otherwise let $h = -1$. Conditioned on assumption (3) that the noise is never large, the "noisy" difference $\widehat{compare}_t - \widehat{thresh}_t$ and the "true" difference $(|answer_h - answer_t|) - thresh$ differ by at most 2α (see the definition of α in Figure 2). This means that in every heavy round t, the difference between $answer_h$ and $answer_t$ is at least d. Since, by Assumption (2), **Alg** is (k, d)-unvarying, we conclude that the with all but $(\beta + \beta_0 + \gamma)$ probability, the algorithm does not terminate by the HALT instruction Step 3b.

Given that the algorithm never terminates in Step 3b, what is its error? Still under our assumptions (1) and (3) that **Alg** is accurate and the noise is never large, if the comparison of Step 3 evaluates to "true," so the round in light, we have that $out_t = out_{t-1} = \cdots = out_h$ is within a $d + 4\alpha$ error of $\mathbf{Alg}_t$. If t is heavy, then out_t is within an α error of $\mathbf{Alg}_t$. We conclude that with all but $(\beta + \beta_0 + \gamma)$ probability, the error in all rounds (simultaneously) is indeed at most $(d + 4\alpha)$ plus the error of **Alg**, i.e. the total error is at most:

$$d + \alpha_0 + (4\rho \cdot (\log(1/\beta) + \log T) \cdot (k + \log^2 T)/\varepsilon).$$

Privacy. To argue pan-privacy, we first use the ε_0 pan-privacy of **Alg** (and the composition of differential privacy) to argue that **Alg**'s internal state during an intrusion is privacy preserving. The proof of privacy for the combination of the internal state and the *times* of updates, *i.e.*, the set of the heavy rounds, is similar in spirit to the proof of pan-privacy for the counter. Given this, and using composition of differential privacy, the actual updated outputs in the at most k heavy rounds will only incur an additional ε-cost in privacy.

Fix two adjacent databases x and x'. Fix an arbitrary execution E_x when the input stream is x. This amounts to fixing the randomness of the algorithm, which in turn fixes the noise values generated. We will describe a corresponding

Transformation for Alg with parameters $(T, \varepsilon, \beta, \rho, k, d, \gamma)$

Initialization. Initialize $\xi_{thresh} = \rho \cdot \log T/\varepsilon$, $\xi_{compare}, \xi_{answer} = 2\rho \cdot (k+1)/\varepsilon$, $m = 0$, $answer_{-1} = 0$, accuracy $\alpha = \rho \cdot (\log(1/\beta) + \log T) \cdot (k + \log^2 T)/\varepsilon$ and $thresh = d + 2\alpha$.

Noise. At time period $0 \le t < T$, choose noise values $\eta_{compare,t} \sim Lap(\xi_{compare})$ and $\eta_{answer,t} \sim Lap(\xi_{answer})$. We also associate $\log T$ threshold noise values with every time period t. To do this, view $t \in [T]$ as a string in $\{0,1\}^{\log T}$. For the i-bit prefix $t_{|i}$ of t (where $i \ge 1$), sample the noise value $\eta_{thresh,t_{|i}} \sim Lap(\xi_{thresh})$.

Persistent State. The value $\eta_{thresh,t_{|i}}$ is sampled and stored in round $(t_{|i} \circ 0^{\log T - i})$, and is erased by the end of round $(t_{|i} \circ 1^{\log T - i})$. There are $2T - 2$ of these noise values in total. The values $\eta_{compare,t}$ and $\eta_{answer,t}$ are sampled in round t and erased by the end of round t; that is, they do *not* persist.

Processing. For $t \in \{0, 1, \ldots, T-1\}$, let $answer_t$ denote the answer computed by **Alg** at time t (recall that we enumerate time and input symbols starting with 0). inputs

1. $\widehat{compare}_t \leftarrow |out_{last} - answer_t + \eta_{compare,t}|$
2. $\widehat{thresh}_t \leftarrow thresh + \sum_{i=1}^{\log T} \eta_{thresh,t_{|i}}$
3. If $\widehat{compare}_t \le \widehat{thresh}_t$ then $out_t \leftarrow out_{t-1}$ else DO:
   (a) $out_t \leftarrow answer_t + \eta_{answer,t}$
   (b) If $m < k$ then $m \leftarrow m + 1$ else HALT.

Figure 2: Continual Output Transformation for Algorithm Alg

execution $E_{x'}$ by describing how its noise values differ from those in E_x. The adversary's view in $E_{x'}$ will be identical to its view in E_x; moreover the probabilities of corresponding noise values differ only by an $e^{3\varepsilon}$ ratio. This implies 3ε differential pan-privacy.

Let η be the noise vector in execution E_x. The adversary's view consists of the incidence vector v for the set of heavy rounds (these are the rounds in which the output changes), the values of the heavy rounds' outputs, and the internal noise values seen in the intrusion. Let the intrusion occur immediately following round t^* of E_x. Then the observed noise values are a subset of $\{\eta_{thresh,t^*_{|i}}\}_{i=1}^{\log T}$. In constructing the noise vector for $E_{x'}$ we will only change noise values *other than* the threshold noises in round t^*; thus we only change noise values that the are *not* observed during the intrusion.

First, to ensure that all light rounds in E_x are also light for in $E_{x'}$, we want to increase the sum of threshold noises in every round except t^* by ρ. This can be done by increasing the values of $\log T$ threshold noises as described below. For $t \in \{0, 1, \ldots, T-1\}$, and $i \in \{1, \ldots, \log T\}$, we view the noise value $\eta_{thresh,t_{|i}}$ as "covering" or "corresponding to" the segment of $2^{\log T - i}$ time periods $\{t_{|i} \circ 0^{\log T - i}, \ldots, t_{|i} \circ 1^{\log T - i}\}$.

For every round t^*, there exists a set S^* of exactly $\log T$ segments, such that every time period *except* t^* is covered by exactly one of these segments. To see this, we start with S^* being empty, and then for each i going from 1 to $\log T$, we choose the one segment of size $2^{\log T - i}$ that isn't covered by previously chosen segments and does not cover t^*. In each of these iterations we cover half of the remaining segments, and all of the segments we choose are disjoint. This means that after $\log T$ iterations, every time period except t^* is covered by exactly one segment. Now in η' we increase the noise values of each of these segments in S^* by ρ. This guarantees that when running with noise η' the sum of threshold noises in each round except t^* is larger by ρ than in the run on x (with noise η). In particular, since the sensitivity of **Alg** is at most ρ, with the possible exception of round t^*, light rounds in E_x with noise η will also be light in $E_{x'}$ with noise η'. Because the threshold noises were chosen from the Laplace distribution with magnitude $\rho \cdot \log T/\varepsilon$, the probabilities of these new values differ by at most an e^{ε} factor from the original noise values.

Now to ensure that heavy rounds in E_x are also heavy in $E_{x'}$, we need to change the noise $\eta_{compare,t}$ by at most 2ρ between η and η' for every heavy round t: we may need a change of ρ to handle the sensitivity of **Alg**, and another change of ρ to handle the increased sum of noisy thresholds. Note that the decision as to whether the noise is increased or decreased depends on $\mathbf{Alg}_t$ and the noise in previous rounds, but this dependence is fixed and so indeed we have a bijection between the vector of noise values. There are at most k heavy rounds in any execution, and so E_x has at most k heavy rounds. We choose the comparison noises from a Laplace distribution with magnitude $2\rho \cdot (k+1) \cdot \varepsilon$. This change in the noise values changes the noise probabilities by at most an $e^{\varepsilon k/(k+1)}$.

Finally, to ensure that if round t^* is light in E_x then it is also light in $E_{x'}$, we may change the comparison noise $\eta_{compare,t^*}$. This poses no risk to privacy as this value does not persist after round t^*, and so will not be seen during the intrusion.

Thus, we have obtained an identical incidence vector by changing the noise probabilities by at most an $e^{2\varepsilon}$ factor. To argue privacy for the actual answers generated in the heavy rounds, we simply note that there are at most k such answers and this incurs another ε cost in privacy. In total, the transformed algorithm is $(\varepsilon_0 + 3\varepsilon)$-differentially pan private with continual outputs. If the original algorithm **Alg** was not pan-private, then the transformed one is not pan-private either, but its collection of outputs is still 3ε-differentially private. □

5.1 Lower Bounds for User-Level Continual Output Privacy

In the previous section, we saw that it is possible to accurately approximate a wide class of functions with user-level privacy even under continual observation (and moreover some of these algorithms are even pan-private). In par-

ticular, this was the case for unvarying functions. In contrast, consider a highly varying function such as modular incidence counting, where we want to estimate the number (between 0 and n) of users that appear an odd number of times (or more generally i times $\pmod k$). Each occurrence of each user is a significant event. Over many time periods, the value of this function can vary wildly many times. Thus, the general transformation of Section 5 does not yield a privacy-preserving algorithm. This is no accident. Indeed, the following theorem shows that there is no user-level differentially private continual-output algorithm for any highly varying function. (See Definition 5.2 for the definition of unvarying algorithms.) As this includes modular incidence counting, we obtain a natural statistic that can be computed pan-privately (see [13]), but that cannot be computed privately in the continual output setting (even without pan-privacy).

DEFINITION 5.3. *Let f be a function defined over a database with n users, with the property that f on a blank symbol does not change value. f is (k,d)-varying if there exists an input y and k indices, $t_1 < t_2 < \ldots < t_k$, such that for all i, $1 \leq i \leq k$, $|f(y_{t_i}) - f(y_{t_{i+1}})| \geq d$. That is, the function f on y jumps by at least d units k times.*

THEOREM 5.3. *Let f be a (k,d)-varying function defined over a database with n users. Then every differentially private, continual output algorithm must have additive error $\Omega(dk/n)$ (even with $\varepsilon = 1$).*

PROOF SKETCH.. We assume throughout that the global sensitivity is $\rho = O(dk/n)$. Otherwise, if $\rho = \Omega(dk/n)$, then by a standard argument for ε-privacy the error must be $\Omega(dk/n)$.

First we claim that if f is (k,d)-varying, then there exists a set S of 2^k inputs such that for every pair x, y of inputs from S, there exists an index i such that $|f(x_1,\ldots,x_i) - f(y_1,\ldots,y_i)| \geq d$. Let y be the input such that f jumps by d at least k times. That is, there exist indices $t_1,\ldots,t_k$ such that $|f(y_{t_i}) - f(y_{t_{i+1}})| \geq d$. We partition y into k intervals, $I_1,\ldots,I_k$, as follows. The interval I_l begins at $t_{l-1}+1$ and ends at t_l. Each input $v \in S$ will consist of k intervals, and the total length of each v will be $2|y|$. Letting $v(\alpha_1,\ldots,\alpha_k)$ denote a particular string in S, where $\alpha_i \in \{0,1\}$, the i^{th} interval of $z(\alpha_1,\ldots,\alpha_k)$ is defined as follows: (i) if $\alpha_i = 0$, then the i^{th} interval will contain a copy of I_i followed by $|I_i|$ blank symbols; (ii) otherwise if $\alpha_i = 1$, then the i^{th} interval will contain $|I_i|$ blank symbols followed by a copy of I_i. This defines a set of 2^k inputs.

Now consider two distinct inputs $v(\alpha), v(\beta)$ from S. We want to show that at some location l, the output of f after reading the l^{th} bits of $v(\alpha)$, $v(\beta)$ differs by at least d. By construction, $v(\alpha)$ and $v(\beta)$ must differ in at least one interval. Suppose it is interval j, so that the j^{th} interval of $v(\alpha)$ consists of I_j followed by an equal number of blanks, and $v(\beta)$ consists of $|I_j|$ blanks followed by I_j. At the location immediately preceding the j^{th} interval of $v(\alpha)$, the value of f on $v(\alpha)$ and $v(\beta)$, is the same, and is equal to the value of f on the first $j-1$ intervals of y. At the first location in the j-th interval of $v(\alpha)$, the value of f jumps by d, whereas the value of f on $v(\beta)$ on the same length substring has not changed from its previous value. Thus the claim is proven.

Secondly we note that if f is (k,d)-varying, then f is also $(t, dk/t)$-varying, as long as $\rho \leq dk/2t$. If the sensitivity is ρ, then for any $a < d/2\rho$, we can split up an interval with a "jump" (change in the function value from the interval's beginning to its end) of d, into a subintervals where each subinterval has a jump of at least d/a (this uses the upper bound on the function's sensitivity). Setting $a = t/k$, which is less than $d/(2\rho)$ as long as $\rho \leq dk/2t$, we end up with t intervals in total, each with a jump of at least dk/t.

Now let S be the set of 2^t inputs, with the property that for every pair of inputs in S, there exists a time period where the output of f differs by at least $d' = dk/t$. If A is accurate, then w.h.p. its output is within $d'/2$ of the correct answer in every time period. If we associate with each input in S an event, corresponding to the algorithm being with $d'/2$ of the correct answer on that input in every round, then this set of events is disjoint. By privacy, however, each of these events has some (exponentially small in n) probability on every input. This implies a contradiction, since there are $|S| = 2^t$ disjoint events, and for $t > n$ the sum of these probabilities is larger than 1. See the full version for details (there we also rule out even algorithms that are accurate in a large fraction of the rounds). □

Note that the lower bound here is not tight with the upper bound of Theorem 5.2. Finding a tight characterization is an intriguing direction for future work.

6. HOW TO USE EXPERT ADVICE

The goal of this section is demonstrating that continual observation counters can be applied to obtain privacy preserving solutions to fairly complex problems, especially in online environments. There are several algorithms in various settings that rely on counting a number of "successes" or "accesses", determining their action based on this count. An underlying principle behind many such algorithms is aggregating the advice of several experts, and choosing a response that is not much worse than any of the experts (regret minimization). In this section out goal is showing that such algorithms can be utilized *without revealing the advice given by the experts*. I.e., we protect the experts' privacy. The main issue is whether the accuracy of the approximate counter is good enough for the application considered. In this extended abstract we focus on the problem of combining expert advice (in the full version we also consider problems such as list access or list update, and how to run many different counters simultaneously).

We present a method for combining expert advice that is competitive with the best expert in hindsight. The setting is that there is a sequence of actions by n experts. At each round each expert decides on an action, and following the round the cost associated with the action is revealed (it is a value in $[0,1]$). The combiner has to choose one of the experts at each round, and act by following that expert, where all the information the combiner has is the cost of the previous actions by the experts. The goal is to minimize *regret*, that is to compare the combiner's performance to the best expert in hindsight. Known performance guarantees for this problem (without privacy concerns) are of the form $(1+\gamma)(\text{best expert}) + O(\log n/\gamma)$, for any γ. That is, there are combiner strategies that only incur an $(1+o(1))$ multiplicative overhead over the best expert in hindsight.

The problem has a long history, see the survey by Blum and Mansour [4]. Our goal is performing the same sort of aggregation, but while protecting the privacy of each expert's action with respect to an observer that sees which expert is

being followed. That is, for any time period t and expert i we guarantee ε-differential privacy for expert i's decision in step t. Here we aim for *event level* privacy.[6]

Most of the combining algorithms in the literature rely, when making a decision at the t^{th} round, on the costs incurred by the experts so far, and not, say, on when exactly these costs occurred. This suggests applying privacy-preserving counters to this problem. The issue, however, is showing that the *approximate* estimation of the cost given by the privacy-preserving counter still serves the combiner of the experts problem (almost) as well as an exact computation of the cost. Some of the well known combining algorithms, like Randomized Weighted Majority, use multiplicative weights or more involved functions of the cost, making this a non-trivial obstacle.

Kalai and Vempala [18] suggested a very simple additive algorithm called *Follow the Perturbed Leader*: For each round $t = 1, 2, \ldots, T$: (1) for each expert $i \in [n]$ pick weight $p_t[i]$ independently chosen according to the exponential distribution with parameter γ, and (2) for each expert i let $C_t(i)$ be the total cost so far. Pick the expert minimizing $C_t(i) - p_t(i)$ and act accordingly.

For this algorithm, as well as Randomized Weighted Majority, the regret (w.r.t the best expert in hindsight) is bounded by $\gamma(\text{best expert}) + \log n/\gamma$. The natural way to adapt this algorithm is to have, for each expert, a private counter counting the cost incurred so far (it is easy to adapt the counter to work with continuous, rather than discrete, increments).

The problem is that we do not have access to the real cost, rather we have access to an approximation C'_t where we are guaranteed to have $C_t(i) \in C'(t) \pm \log^{1.5} t$ (w.h.p in each round). A possible approach to the problem is to show that this is true for *any* adversarial approximation, that is at each point in time the estimate C' of the cost so far is guaranteed to be with Δ of the real cost C, but an adversary may set the approximation as it wishes within these bounds, and the question is what combining algorithm is appropriate. It is not clear whether such a general approach will work.

Suppose, instead, that the approximation works by adding noise, where at any point in time the distribution of the noise is identical. This, in fact, is the case for the counter algorithm of Section 4: in each round, the noise is the sum of (the same number of) identically distributed Laplace variables. Consider the cost of the expert combining algorithm. It is equal to $\sum_{t=1}^{T} Y_t$ where Y_t is a random variable representing the cost of the combining algorithm at step t. The r.v's $\{Y_t\}$ depend on the noise added at each step and the randomness of the combining algorithm (the perturbations in the Kalai-Vempala algorithm). From linearity of expectation $E[\sum_{t=1}^{T} Y_t] = \sum_{t=1}^{T} E[Y_t]$. The expectation of the random variables Y_t remains the same when the algorithm is run on counters where exactly the same noise is added at all the steps: that is, the noise is generated only for one round and then *exactly the same noise* is added at all rounds. If this is the case (i.e. the same noise added at all rounds), then we can think of the algorithm as running with a prefix of steps where the cost of each expert in that prefix equals the noise added to it. Now take the guarantee on the regret w.r.t to the best expert. The difference between the real leading expert and the leading expert with the approximation noise is at most the maximal noise. So we get that the new combining algorithm is within the bound the original algorithm was plus the maximum of n random variables, each a sum of $(1+\log T)$ Laplacians with magnitude $\epsilon/\log T$, that is an additional $O(\log n \cdot log^{1.5} T/\epsilon)$ term to the regret. Note that here we use the fact that in each round individually, w.h.p. the noise added by the counter has magnitude $O(\log^{1.5} T/\varepsilon)$ (see the discussion following Theorem 4.1).

[6] In fact, it can be shown that it is impossible to obtain any non trivial result with respect to user level privacy.

7. REFERENCES

[1] J. L. Bentley and J. B. Saxe. Decomposable searching problems i: Static-to-dynamic transformation. *J. Algorithms*, 1(4):301–358, 1980.

[2] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: The SuLQ framework. 24th PODS, 2005.

[3] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of STOC*, 2008.

[4] A. Blum and Y. Mansour. Learning, regret minimization, and equilibria. In *Algorithmic Game Theory*. Cambridge University Press, 2007.

[5] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proceedings of 22nd PODS*, pages 202–210, 2003.

[6] C. Dwork. A firm foundation for private data analysis. *Communications of the ACM (to appear)*.

[7] C. Dwork, Differential privacy in new settings. *Proc. ACM-SIAM SODA*, 2010.

[8] C. Dwork. An ad omnia approach to defining and achieving private data analysis. In *Privacy, Security, and Trust in KDD, First ACM SIGKDD International (PinKDD), Revised Selected Papers*, volume 4890 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2007.

[9] C. Dwork. The differential privacy frontier. In *Proceedings of TCC*, 2009.

[10] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: privacy via distributed noise generation. In *Advances in Cryptology: Proceedings of EUROCRYPT*, pages 486–503, 2006.

[11] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of TCC*, pages 265–284, 2006.

[12] C. Dwork, F. McSherry, and K. Talwar. The price of privacy and the limits of lp decoding. In *Proceedings of STOC*, pages pp. 85–94, 2007.

[13] C. Dwork, M. Naor, T. Pitassi, G. N. Rothblum, and S. Yekhanin. Pan-private streaming algorithms. In *Proceedings of ICS*, 2010.

[14] C. Dwork, M. Naor, O. Reingold, G. Rothblum, and S. Vadhan. When and how can privacy-preserving data release be done efficiently? In *Proceedings of STOC*, 2009.

[15] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *Proceedings of CRYPTO 2004*, volume 3152, pages 528–544, 2004.

[16] C. Dwork and S. Yekhanin. New efficient attacks on statistical disclosure control mechanisms. In *Proceedings of CRYPTO 2008*, pages 468–480, 2008.

[17] M. Hardt and K. Talwar. On the geometry of differential privacy. arXiv:0907.3754v2, 2009.

[18] A. T. Kalai and S. Vempala. Efficient algorithms for online decision problems. *J. Comput. Syst. Sci.*, 71(3):291–307, 2005.

[19] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Proceedings of FOCS 2007*.

[20] S. Warner. Randomized response: a survey technique for eliminating evasive answer bias. *JASA*, pages 63–69, 1965.