Pan-Private Streaming Algorithms

Cynthia Dwork[1] Moni Naor[2,*] Toniann Pitassi[3,†] Guy N. Rothblum[4,‡] Sergey Yekhanin[1]
[1]Microsoft Research, Silicon Valley Campus
[2]Department of Computer Science and Applied Math, Weizmann Institute of Science
[3]Department of Computer Science, University of Toronto
[4]Department of Computer Science, Princeton University
dwork@microsoft.com moni.naor@weizmann.ac.il toni@cs.toronto.edu rothblum@mit.edu
yekhanin@microsoft.com

Abstract: Collectors of confidential data, such as governmental agencies, hospitals, or search engine providers, can be pressured to permit data to be used for purposes other than that for which they were collected. To support the data curators, we initiate a study of *pan-private* algorithms; roughly speaking, these algorithms retain their privacy properties even if their internal state becomes visible to an adversary. Our principal focus is on streaming algorithms, where each datum may be discarded immediately after processing.

Keywords: Differential Privacy, Pan Privacy, Streaming Algorithms

1 Introduction

Data collected by a curator for a given purpose are frequently subject to "mission creep" and legal compulsion, such as a subpoena. For example, in the context of professional baseball, urine samples, originally collected for one purpose – a statistical study of drug use in Major League baseball – were later used for a different purpose: criminal investigation of the Bay Area Laboratory Co-Operative, eventually resulting in a privacy breach for 104 players who tested positive. The point here is not to support drug use in professional sports. Rather, that statistical studies are frequently conducted for anticipated societal gain; our goal is to facilitate these studies by designing technology that encourages participation.

Any data curator – a library, a clinic, the US Census Bureau, a social networking site, or a search engine provider – can be pressured to permit data to be used for purposes other than that for which they are collected. When, as in the study of drug use in professional baseball, the purpose of the collection is for statistical data analysis, there may be no need to store the data once the analysis is complete. More generally, if the analysis is ongoing, as in the monitoring of a sensor network, obsolete status information may reasonably be discarded as new information is gathered. This suggests an investigation of privacy-preserving streaming algorithms with small state – much too small to store the data – and indeed the results reported here do focus on the streaming model. However, nothing in the definition of a streaming algorithm, even one with relatively small state, precludes storing data pertaining to certain individuals of interest. Popular techniques from the streaming literature, such as Count-Min Sketch and subsampling, do precisely this. As we argue below, this is not satisfactory from a privacy point of view. A subpoena or other intrusion into the local state will breach the privacy of the sampled individuals.

We therefore initiate a study of *pan-private* algorithms; roughly speaking, these algorithms retain their privacy properties even if their internal state becomes visible to an adversary. We stress that, just as a streaming algorithm may fail to be pan-private, a batch algorithm may be pan-private, yielding protection against, say, an intruder that cannot access the data, perhaps because they are stored on a different server. Nonetheless, we focus here on pan-private streaming algorithms, so that the data can be discarded. Our notion of privacy is differential privacy [1, 2]. It may seem that without some kind of "secret state" it will be impossible to achieve accurate yet private calculations. We show this is not the case: there is quite a rich set of functions that can be computed by means of pan-private streaming algorithms.

Results. Our contributions are definitional and algorithmic, as well as in showing impossibility results. We introduce two new notions: *pan privacy* and *user-level privacy*. We differentiate between privacy at the user level of granularity (pure differential privacy) and at the level of granularity of individual events (much weaker). We also consider a very strong versin of pan-privacy, in

*Research supported in part by a grant from the Israel Science Foundation. Part of this work was done while visiting Microsoft Research and Princeton University.

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

‡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 Microsoft Research.

which the algorithm is immune to continual intrusion on the internal state.

We provide pan-private streaming algorithms for several counting tasks: estimating the number of distinct elements in a stream, estimating cropped means, estimating the number of heavy hitters, and estimating frequency counts ("How many items appear in the stream exactly k times?"). As already noted, sampling items from the input stream, a common technique in streaming algorithms, is problematic for pan-privacy. For example, although the probability of selecting a specific individual may be small, once a record belonging to this individual has been stored, it is implausible to deny that the individual's data appear in the data stream. To appreciate the strength of the guarantee that we provide ("user level privacy"), consider, for example, analyzing click stream and query log data. The data in this source are generated by individuals, and we wish to assure their privacy. For this we require that any two streams, one with and the other without all information of a particular individual, will produce very similar distributions on states and outputs, even though the data of an individual are interleaved arbitrarily with other data in the stream. (This interleaving complicates both privacy and accuracy.) The pan private algorithms are all single pass and space efficient and the output is ε-differentially-private at the user level. Our algorithms address the following problems:

- Estimating the fraction of elements in X that appear at least once in the data stream. The algorithm requires space $\mathrm{poly}(1/\varepsilon, 1/\alpha, \log(1/\beta))$ and guarantees ε-differential pan-privacy; with probability $1-\beta$ it provides accuracy α.[1] This is done by an adaptation of the randomized response technique for this environment.
- Approximating for any t the fraction of elements in X that appear exactly t times in the data stream (the t-*incidence* items). This is done by a development of a hashing tool relevant for this problem, which is a t-*modular incidence* count.
- Estimating what we call the t-*cropped mean*: roughly, the average, over all users, of the minimum of t and the number of times the user appears in the data stream. This is done using a randomized initial shift, together with the idea of randomized response. Combining these ideas, each item appearing less than t times contributes *in expectation* its number of appearances to the cropped mean, and each item appearing more than t time contributes t (in expectation).
- Estimating the fraction of k-heavy hitters (elements of X that appear at least k times in the data stream). This is our technically most challenging result; it makes extensive use of the cropped mean algorithm. The notion of utility achieved is slightly relaxed: the answers are approximately accurate for a stream $\mathcal{S}'$ that is *close* to the input stream $\mathcal{S}$: the numbers of appearances of *every* item in $\mathcal{S}$ and $\mathcal{S}'$ differ by at most a small multiplicative factor.

We give several impossibility results. We provide a strong impossibility result for *finite state* pan-private streaming algorithms against even *two* unannounced intrusions for the density estimation problem (our algorithms all require only finite space). We also show that, for the same problem, no pan-private algorithm (even non finite space) that guarantees privacy against continual intrusions can provide ε-differential pan-privacy with sublinear error. Finally, we place pan-private streaming algorithms in context, separating them from *randomized response* [3] and *private sketching* [4]. That is, we show problem for which there exists an algorithm immune to continual intrusions in our streaming model, but no such algorithm exists in the sketching model (where the influence of each input is completely independent of the state).

1.1 Related Work

There is a vast literature on (non-private) streaming algorithms, and summarizing it is beyond the scope of this extended abstract. See, for example, [5] and the references contained therein, the recent course notes [6] and work on streaming algorithms in the machine learning literature, e.g. [7]. The fraction of the universe X of items that appears at least once in the data stream is analogous to the number of distinct items in the stream, which is a well studied measure issue in streaming. Datar and Muthukrishnan [8] motivate and study the problem of approximating the number of α-*rare* elements – i.e., elements appearing exactly α times – in the so-called "windowed" streaming model. (We call this "incidence" to be clear that we mean an exact number of times and not a lower or upper bound.) Our cropped mean quantity was previously studied (without a name) by Indyk [9], who used it in constructing streaming algorithms for facility location problems.

There is also a vast literature on private data analysis, both in statistics and in computer science. The notion of *differential privacy* is due to Dwork, McSherry, Nissim, and Smith [1, 2]. Most of the work on private data analysis assumes a trusted curator. The problem of finding privacy-protective methods in order to encourage survey respondents to respond to sensitive questions (involving, say, sexuality or criminal behavior) – essentially, private data analysis with an untrusted curator – was first studied by Warner [3]; modern incarnations include randomizing market baskets [10] and private sketches [4][2]. With certain technical changes

[1] Though our main theorem statements are for additive accuracy, all results can be modified to give multiplicative accuracy. See the discussion towards the end of Section 3 for details.

[2] This is a different use of the word "sketch" than is in the stream-

(involving the use of ε-biased hash functions instead of the pseudo-randomness used in the original work), the sketching techniques can be used to give a weak form of pan-private algorithms for estimating the number of times an item appears in a stream. This can also be achieved using Count-Min Sketch [5].

There is some work related to privacy regarding problems typically studied in the streaming literature. In this work the scenario is that the data are split between two centers who do not wish to reveal the value of their inputs beyond what is necessary ("two-party secure computation") [11, 12].

Several of the classical streaming problems related to counting have natural differentially private (but not pan-private) implementations, obtained by simply adding appropriately generated noise. Recent work of Feldman, Fiat, Kaplan, and Nissim addresses creation of differentially private coresets [13]. As noted in [13], coreset constructions frequently imply streaming algorithms, and indeed this fact, combined with results of Feldman *et al.*, can yield differentially private streaming algorithms; these algorithms are not pan-private.

A concern reminiscent of pan privacy was explored in the area of data structures under the name of *history-independence* [14, 15]. The goal there was to design data structures for which the internal representation of the data structure does not reveal more than what the standard interface reveals.

2 Definitions

We assume a data stream of unbounded length composed of elements in a universe X. It may be helpful to keep in mind, as motivation, data analysis on a query stream, in which each item is a query accompanied by the IP address of the issuer. For now, we ignore the query text itself; the universe X is the universe of potential IP addresses. Thus, intuitively, privacy protects the presence or absence of an IP address in the stream, independent of the number of times it arises, should it actually be present at all.

Recall that we are interested in Differential Privacy. In the "batch processing world" this meant:

Definition 2.1. [1] A randomized function $\mathcal{K}$ gives *ε-differential privacy* if for all *adjacent* (differing in at most one row) data sets D and D', and all $S \subseteq Range(\mathcal{K})$,

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

where the probability space in each case is over the coin flips of $\mathcal{K}$.

We need to update the definition for our purposes, given that we work with data streams, in particular with click streams where the information regarding a user is spread all over the stream.

X-Adjacent Data Streams and Differential Privacy. An important point we make regarding the privacy requirements is related to the notion of adjacency, i.e. what does it mean for D and D' in Definition 2.1 to be adjacent when they are data streams. We distinguish between "event-level" and "user-level" privacy of data streams. In the above example of a click stream, event level means that given the output of the function it is hard to determine whether a particular query from a given IP occurred. User level provides a much stronger guarantee to users: it is hard to determine whether a given user's (IP address) was ever included in the stream at all. We formalize this notion below. The algorithms we propose all offer user-level privacy.

Definition 2.2. Data streams (or stream prefixes) S and S' are *X-adjacent* if they differ only in the presence or absence of *any number of* occurrences of a single element $x \in X$. In other words, if all occurrences of x are deleted from both streams, then the results should be identical.

Outputs. An algorithm runs until it receives a special signal, at which point it produces outputs. Thus, the algorithm does not know the length of the stream in advance. The algorithm may optionally continue to run and produce additional outputs later, again in response to a special signal.

What Can be Observed by the Adversary, and What Cannot. An algorithm moves through a sequence of internal states and produces a (possibly unbounded) sequence of outputs. We assume that the adversary can only observe internal states and outputs; the adversary cannot see the data in the stream (although it may have auxiliary knowledge about some of these data). We can model this through the assumption that in a single atomic step the algorithm can read a datum, compute, and change the internal state. Some sort of granularity assumption is essential, since otherwise the adversary could learn the raw data and privacy is impossible.

Adversary Capabilities. Our algorithms should maintain their accuracy and privacy guarantees in the worst case, provided (as usual, in the streaming literature) that these worst-case inputs are chosen *non-adaptively*, independent of the observed internal states and outputs. In fact, privacy holds unconditionally, even against an adaptive adversary who chooses the stream as a function of the observed internal states and outputs[3]. The adversary may have arbitrary auxiliary knowledge obtained from other sources (this captured by the notion of differentially privacy and the fact that we consider

ing literature; here it refers to a randomly chosen object that, when concatenated with an individual's data, hashes to a desired value.

[3] Unsurprisingly, many of our algorithms can be modified to simultaneously retain privacy and accuracy even against an adaptive adversary, but the space requirements may be excessive.

X-adjacency). It can also have arbitrary computational power.

We now define the basic requirement of pan privacy, which is essentially immunity to a single intrusion. As explained, the extension to multiple intrusions is straightforward.

Definition 2.3. Pan-Privacy (against a single intrusion).

Let $\mathbf{Alg}$ be an algorithm. Let I denote the set of internal states of the algorithm, and σ the set of possible output sequences (the output could be empty, for instance if it is at the start of the sequence). We think of algorithm $\mathbf{Alg}$ as a pair of probabilistic mappings: $\mathbf{State}$ maps data stream prefixes to a state (the one that the algorithm is in immediately after processing the prefix), and $\mathbf{Out}$ maps prefixes to an output sequence (the one produced while processing the prefix). Then algorithm $\mathbf{Alg}$ is *pan-private* if for all sets $\mathrm{I}' \subseteq \mathrm{I}$ and $\sigma', \sigma'' \subseteq \sigma$, and for all pairs of X-adjacent data stream prefixes S and S the following holds. Let $x \in X$ be the unique single symbol such that S and S' differ on the number and or placement of instances of x. For any U and U' also differing only on instances of x, such that $S = UV$ (that is, S is the concatenation of U and V), and $S' = U'V'$,

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

where the probability spaces are over the coin flips of the algorithm $\mathbf{Alg}$.

The definition requires that the joint distribution on internal states and outputs be essentially the same after processing adjacent prefixes, but it is stronger. For example, the definition captures the case that intrusions happen immediately following adjacent prefixes and, possibly some time later, additional output is produced. To extend this to multiple intrusions, one considers adjacent streams $S = U_1 \ldots U_d V$ and $S' = U'_1 \ldots U'_d V'$, where for $1 \leq i \leq d$ it holds that $U_1 \ldots U_i$ and $U'_1 \ldots U'_i$ are adjacent, differing only in the symbol on which S and S' differ.

The case of even only two un-announced intrusions, and even more so the case of continual intrusions, are both hard to deal with. Negative results to this effect appear in Section 6.

Remark 2.1. *Several remarks are in order.*

(1) If we assume the existence of a very small amount of secret storage (not visible to the adversary), then many problems for which we have been unable to obtain pan-private solutions have (non-pan) private streaming solutions. However, we don't see the amount *of secret storage to be so important as its* existence*, since secret storage is vulnerable to the social pressures against which we seek to protect the curator.*

(2) If this issue could be addressed, then it should be possible to design a rich class of algorithms based on homomorphic encryption schemes, where the decryption key is kept in the secret storage. Pan-privacy would then hold against computationally bounded adversaries.

(3) If the adversary is computationally bounded, the algorithms' processing stages can be made deterministic. To do this, we generate the needed randomness using a forward-secure *pseudo-random number generator [16]. This ensures that even when an intrusion occurs, the randomness used in the past still looks random to the intruder. Provided the intrusion is announced, the randomness can be refreshed and pan-privacy continues to hold. Many classical pseudo-random generators have this property, e.g. [17, 18].*

3 Density Estimations and Cropped Means

The *density* of the input stream is the fraction of items in X that appear at least once in the stream. In this section we describe pan-private streaming algorithms for estimating the density of the input stream and a newly defined generalization, which we call the t-cropped mean. This is the mean, over all items in the universe X, of (for each item) the minimum between that item's incidence count in the stream (the number of times it appears) and the value t.

We first concentrate on *additive* errors, and then describe briefly how to modify the algorithm(s) to obtain multiplicative error.

We begin with the density estimation algorithm. Both algorithms use a technique based on *randomized response*. The difficulty is that we aim for *user level* privacy. For accuracy, the algorithm has to maintain information regarding previous appearances of items, so as to interpret new appearances correctly. But for user-level privacy, this should be done without the internal state leaking significant information about the appearance or non-appearance of individual items.

First, in order to keep the state small, we sample a random a set $M \subseteq X$ of m items, where m is large enough so that, with high probability, the density of M's items in the input stream approximates the density of X's items.

The algorithms maintains a table, with one entry per item in M. For items in M that have not yet appeared, their entry will be drawn from a distribution $\mathcal{D}_0$ (over $\{0, 1\}$). For items that have appeared at least once, their entry will be drawn from $\mathcal{D}_1$ *no matter how many times they have appeared*. These two distributions should be "close enough" to guarantee pan-privacy for individual users, but "far enough" to allow collection of aggregate statistics about the fraction of users that appear at least

once. Specifically, the distribution $\mathcal{D}_0(\varepsilon)$ will give the outputs '0' and '1' both with probability $1/2$. The distribution $\mathcal{D}_1(\varepsilon)$ will give '1' with probability $1/2+\varepsilon/4$ and '0' with probability $1/2-\varepsilon/4$.

The key observation that allows user-level pan-privacy is that if we re-draw an item's entry from D_1 whenever that item appears in the data stream, the distribution of that item's entry is unaffected by its number of appearances (beyond the first). This means that the entry does not yield any information about the item's number of appearances (beyond the first). The algorithm appears in Figure 1.

Claim 3.1. *For $\varepsilon \leq 1/2$, the distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ are "ε-differentially private". For both values $b \in \{0,1\}$, it holds that: $e^{-\varepsilon} \leq \Pr_{\mathcal{D}_1}[b]/\Pr_{\mathcal{D}_0}[b] \leq e^{\varepsilon}$.*

Proof of Claim 3.1. The ratio between the probabilities of 1 by $\mathcal{D}_1$ and $\mathcal{D}_0$ is $1+\varepsilon/2 \leq e^{\varepsilon}$. The ratio between the probabilities of 0 by $\mathcal{D}_1$ and $\mathcal{D}_0$ is $1-\varepsilon/2 \geq e^{-\varepsilon}$ (this inequality holds for $0 \leq \varepsilon \leq 1/2$). □

Theorem 3.2. *Assume $\varepsilon \leq 1/2$. The density estimator of Figure 1 guarantees 2ε differential pan-privacy. For a fixed input, with probability $1-\beta$ over the estimator's coins, the output is α-close to the fraction of items in X that appear in the input stream. The space used is $\mathrm{poly}(1/\varepsilon, 1/\alpha, \log(1/\beta))$.*

Proof of Theorem 3.2. We argue utility and privacy separately.

Privacy. For items not in M no information is stored by the algorithm, so privacy is perfect. For an item in the set M, if the item never appears in the input stream then its entry is drawn from $\mathcal{D}_0$. If the item appears once or more in the input stream, then its entry is drawn from $\mathcal{D}_1$ (multiple appearances only result in multiple samples being taken, but the item's entry in the table remains a sample from $\mathcal{D}_1$). Thus when an intrusion occurs, by Claim 3.1, the users in M are guaranteed ε-differential privacy against the intrusion.

Later, when the algorithm generates its output, the sensitivity of this output to the presence or absence of any one user is at most $1/m$ (or 0 if the user is not in M). By adding noise sampled from $Lap(1/(\varepsilon \cdot m))$ we guarantee that users are guaranteed 2ε privacy against the combined information an adversary can gain from its intrusion and viewing the algorithm's output. We note that we will use the same idea to ensure only an ε added privacy loss from the output throughout this work (we will not repeat the analysis for every algorithm).

Utility. First, we claim that the fraction of items in M that appear at least once is, with probability at least $1-\beta/2$, an $\alpha/2$ approximation to the fraction of items in X that appear at least once in the input stream. This follows by a Chernoff bound, because M is a set of $m \geq \mathrm{poly}(1/\alpha, \log(1/\beta))$ items chosen uniformly and at random from the set X.

We now complete the proof by showing that, with probability at least $1-\beta/2$, the algorithm's output is a $\alpha/2$-approximation to the fraction of items *in* M that appear at least once. Let f be the *true* fraction of items in M that appear at least once. The *expected* fraction of 1-entries in the table (over the randomness of drawing from $\mathcal{D}_0$ and $\mathcal{D}_1$) is then:

$$E[\theta] = f \cdot (1/2+\varepsilon/4) + (1-f) \cdot 1/2 = 1/2 + f \cdot \varepsilon/4$$

We can separate this fraction into the contribution of 1's from items that appeared at least once (samples from $\mathcal{D}_1$) and those that did not (samples from $\mathcal{D}_0$). Taking a Chernoff bound we get that, since $m = \mathrm{poly}(1/\varepsilon, 1/\alpha, \log(1/\beta))$, with probability $1-\beta/2$ the observed fraction of 1's θ satisfies:

$$|\theta - E[\theta]| \leq \alpha \cdot \varepsilon/8$$

Now the algorithm outputs $f' = 4(\theta - 1/2)/\varepsilon + Lap(1/(\varepsilon \cdot m))$. We conclude that with probability $1-\beta$:

$$|f - f'| \leq \alpha$$

□

Obtaining a multiplicative error. If the fraction of elements of X that appear in the stream is very small, an additive accuracy guarantee might not provide a very meaningful result. Instead, in such cases it is better to give a multiplicative guarantee. This can be achieved using the pan-private additive density estimator of Figure 1 as a "black-box". The idea is to hash the universe X into $\ell = \log|X|$ smaller sets $X_0, X_1, \ldots, X_\ell = X$, where set X_i is of size 2^i. The density estimation algorithm is run independently for all ℓ choices of m, using noise $Lap(\ell/\varepsilon \cdot m)$ in the Output step. Finally, the "right" output, among these ℓ outputs, is selected, as described below.

Hashing is done using 4-wise independent hash functions (thus ensuring small space complexity). The density estimator can then be used to obtain an additively accurate estimation γ'_k on the density γ_k of the hash values of input stream items in all of the hash sets. The density of the input stream in a set X_i can give a good idea of the ratio between the number of distinct items in the input stream and $|X_i|$. The only problem is that there may be hash collisions that lead to under-counting, and the number of collisions grows as the set size shrinks.

We thus want to use the density of the "right" X_k, one that is neither too large (leading to the additive error being very large in multiplicative terms) nor too small (leading to many collisions and under-counting). We know that for 4-wise independent hash functions, if the

Density Estimator $(\varepsilon, \alpha, \beta)$

Init. Sample at random a set M of $m = \text{poly}(1/\varepsilon, 1/\alpha, \log(1/\beta))$ elements (representatives) in X. Create a table of size m with a single one-bit entry for each item in M. For every entry $x \in M$, generate a random initial value $b_x \sim \mathcal{D}_0(\varepsilon)$.

Processing. When a value $x \in M$ appears in the data stream, update x's entry in the table by drawing it from $\mathcal{D}_1$: $b_x \sim \mathcal{D}_1(\varepsilon)$.

Output. Compute θ, the fraction of entries in the table with value 1. Output the density value $f' = 4(\theta - 1/2)/\varepsilon + Lap(1/(\varepsilon \cdot m))$.

Figure 1: Density Estimator

density of the output in X_k is γ_k, then with high probability the number of collisions was at most $\gamma_k^2 \cdot |X_k|$ (using Chebyshev's inequality). If we use the first hash function for which $\gamma_k \approx \alpha$ then we get that the number of collisions is w.h.p. bounded by $\alpha^2 \cdot |X_k|$. On the other hand, we know that the number of distinct elements in the input is at least about $\alpha \cdot |X_k|$. If we obtain an α^2-additive approximation to the density of the input's hash in X_k we can multiply it by $|X_k|$ and we actually have an $O((\alpha^2) \cdot |X_k|)$-additively accurate approximation on the *number* of distinct items in the input, which is also an $O(\alpha)$-multiplicative accurate estimation on the number of distinct elements (since the number of distinct elements was at least $\alpha \cdot |X_k|$). This is all w.h.p. over the choice of hash functions (using Chebyshev's inequality), and we can amplify the success probability by choosing several collections of hash functions independently at random and using the median answer, or by choosing hash families with greater independence. (Note that the algorithm implicitly assumes that the number of items in the output universe X_k is at least $m \geq \text{poly}(1/\varepsilon)$ so that we can recover good statistics from the randomized responses.)

We note that this and similar techniques can be used to obtain multiplicative approximations (rather than additive ones) for the other algorithms in this paper. We henceforth confine our attention to additive approximations.

Handling Multiple Announced Intrusions. It is possible to handle multiple *announced* intrusions by re-randomizing the randomized-response bits after each intrusion. For example, after the first announced intrusions, for every $x \in M$ in the table, we re-draw the bit b_x according to its current value: if previously it was 1, we re-draw from $\mathcal{D}_1$, and if it was 0 we re-draw from $\mathcal{D}_0$. Thus entries for items that appeared in the input stream are now distributed according to $\mathcal{D}'_1$, which is 1 w.p. $(1/2+\varepsilon/8+\varepsilon^2/16)$ and 0 w.p. $(1/2-\varepsilon/8-\varepsilon^2/16)$. Entries for items that did not appear in the stream are distributed according to $\mathcal{D}'_0$, which is 1 w.p. $(1/2+\varepsilon/8)$ and 0 w.p. $(1/2-\varepsilon/8)$. We now change the randomized response distributions to be $\mathcal{D}'_1$ (for items that appear) and $\mathcal{D}'_0$ (for items that don't appear). Note that the algorithm keeps track of how many intrusions happened (there is no need to protect the intruder's privacy).

Pan-privacy is as before (the new distributions are even "more private" than the old ones to an attacker that did not see the intrusion). Accuracy, however, does not behave as well: the difference between the probability of 1 in D'_1 and D'_0 shrinks by an $O(\varepsilon)$ factor each time we recover from an intrusion. This means that accuracy degrades very quickly, or (if we want to guarantee accuracy always) that the storage is exponential in the number of intrusions we want to handle.

Similar ideas can be used to handle multiple announced intrusions for other algorithms in this work, with similar degradation in accuracy. We will not revisit this point.

3.1 Estimating the Cropped Mean

For $x \in X$, let n_x denote the number of appearances of x in the input stream. Then the t-cropped mean, for $t \geq 2$, is the quantity:

$$E \downarrow^t = 1/|X| \cdot \sum_{x \in X} \min(n_x, t).$$

This quantity is interesting in its own right, as a measure of the activity level in the input stream. For example, it can be used to estimate the number of items in the input stream while mitigating the effect of outliers who arrive a large number of times. Moreover, we will use our cropped mean algorithm as a central building block block in estimating the fraction of k-heavy hitters (items that appear k times or more).

Our algorithm for cropped mean is a generalization of the density estimation algorithm above. There we maintained a bit ('0' or '1') for every item in a randomly chosen subset M. The bits of items that appeared one or more times all had the same probability of being '1' (regardless of the number of appearances), and this probability was greater than the probability of being '1' for items that never appeared. When estimating the cropped mean we will again choose a random subset of items from X and maintain a bit for each item. The probability of this bit being '1' will grow linearly with

the number of times an item appears, up to t appearances. The bits of items that appear t or more times will all have the same probability of being '1'. Specifically, if an item appears i times, its bit will be '1' with probability $1/2 + \varepsilon \cdot \min(i,t)/4t$, maintaining ε-differential privacy. The fraction of 1's in the resulting table will be used to estimate the t-cropped mean. To implement this table we maintain a counter for each $x \in M$. The counter for an item x will hold (integer) values between 0 and $t-1$. The algorithm initializes the counter to a *random* value in this range, and whenever item x arrives, the counter is increased by 1, modulo t. When the counter increases to the value 0, we sample the bit b_x according to D_1: $b_x \sim \mathcal{D}_1$. Note that the modular counter here is (perfectly) private, and the bit b_x is ε-differentially private (by Claim 3.1). The algorithm appears in Figure 2.

Theorem 3.3. *Assume $\varepsilon \leq 1/2$. The t-cropped mean estimator of Figure 2 guarantees 2ε differential pan-privacy. For a fixed input, with probability $1-\beta$ over the estimator's coins, the output is $(\alpha \cdot t)$-close to the t-cropped mean. The space used is* $\mathrm{poly}(\log t, 1/\varepsilon, 1/\alpha, \log(1/\beta))$.

Proof of Theorem 3.3. The proof of pan-privacy is as in Theorem 1. Note that the $(\mathrm{mod}\ t)$ counters are perfectly private and contain no information about the actual number of appearances made by their items.

We proceed with a proof of utility. First, the t-cropped mean of the items in M is, with probability $1-\beta/2$, at least $(\alpha/2 \cdot t)$-close to the t-cropped mean of all the items in X (by a standard Chernoff bound, since $m \geq \mathrm{poly}(1/\alpha, \log(1\beta))$).

To complete the proof, we need to show that the output f' is, with probability $1-\beta/2$, at least $(\alpha/2 \cdot t)$-close to the t-cropped mean of the items in M:

$$1/m \cdot \sum_{x \in M} \min(n_x, t)$$

For each $x \in M$, which appears n_x times, the probability that b_x is equal to 1 (over the initialization of the modular counter and the sampling from $\mathcal{D}_0$ or D_1) is exactly $1/2 + \varepsilon \cdot \min(n_x, t)/4t$. Thus, using f to denote the *true* t-cropped mean of items in M, then the *expected* fraction θ of 1's in the output table is

$$E[\theta] = 1/2 + \varepsilon \cdot f/4t$$

This is a sum of $\{0,1\}$-random variables, whose expected mean sum is $E[\theta]$. Taking θ' to be the actual fraction of 1's on the table, by a Chernoff bound, since $m \geq \mathrm{poly}(1/\varepsilon, 1/\alpha, \log(1/\beta))$, we get that with probability at least $1-\beta/2$:

$$|\theta' - E[\theta]| \leq \varepsilon \cdot \alpha/8$$

The algorithm outputs $f' = 4t(\theta' - 1/2)/\varepsilon + Lap(t/(\varepsilon \cdot m))$, and by the above inequality we get that, with all but at most β probability:

$$|f - f'| \leq t \cdot \alpha$$

□

4 Estimating the Fraction of k-Heavy Hitters

In this section we present a pan-private algorithm for estimating the fraction of *"k-heavy hitters"*, the fraction (in X) of items that appear k or more times. Note that, unlike the non-private case, we do not try to find *which* items are the heavy hitters, which will be violation of the privacy requirements. Instead, in a pan-private fashion, we estimate the just the fraction of items that are heavy hitters.

We assume that the number of appearances of each item in the data stream is bounded by a (large) constant N. Our complexity will be polylogarithmic in N. Furthermore, as we now explain, in this result we achieve a relaxed notion of utility: the algorithm guarantees that its output is accurate for an input stream $\mathcal{S}'$ that is *close* to the given (original) input stream $\mathcal{S}$ [19]. Let n_x be the number of appearances of item x in the original input stream $\mathcal{S}$, $x \in X$. We say that streams $\mathcal{S}$ and $\mathcal{S}'$ are $(1+\rho)$-*close* if, for accuracy parameter $\rho \in (0,1]$, and $\forall x \in X$,

$$\frac{1}{1+\rho} \cdot n_x \leq n'_x \leq (1+\rho) \cdot n_x$$

The algorithm makes extensive use of the t-cropped mean estimator. The algorithm description and analysis will be for general α-accuracy with $(1+\rho)$-closeness, but in the intuitive exposition α is a constant and we only aim for 2-closeness. We start by estimating the t cropped mean for $t = 1, 2, 4, 8, \ldots, N/2, N$ (assume here N is a power of 2). Let $E \downarrow^t$ denote the t-cropped mean of the input stream. Examine the quantity:

$$E \downarrow^N - E \downarrow^{N/2} = 1/|X| \cdot \sum_{x \in X : n_x \geq N/2} (\min(n_x, N) - N/2)$$

This quantity gives us some idea about the total number of appearances of items that appear more than $N/2$ times. Unfortunately, it does not give us an idea of the *number of distinct items* that appear more than $N/2$ times. For example, if the difference is, say, $z/|X|$, it could be because z items appeared $N/2+1$ times, or it could be because many fewer items, even $2z/N$, appeared N times each. To overcome this problem we turn to the relaxed notion of utility, and only require the answer to be approximately accurate for a "close" input stream.

t-**Cropped Mean Estimator** $(t, \varepsilon, \alpha, \beta)$

Init. Sample at random a set M of $m = \text{poly}(1/\varepsilon, 1/\alpha, \log(1/\beta))$ elements in X. Create a table of size m with a one-bit entry $b_x \in \{0, 1\}$ and a modular counter $c_x \in \{0, 1, \ldots, t-1\}$, for all $x \in M$. In addition, for all $x \in M$, generate random initial values $b_x \in_R \mathcal{D}_0(\varepsilon)$ and $c_x \in_R \{0, 1, \ldots, t-1\}$.

Processing. When a value $x \in M$ appears in the data stream, increase the counter c_x: $c_x \leftarrow c_x + 1 \pmod t$. If the new counter value is 0, then refresh b_x by drawing it from $\mathcal{D}_1$: $b_x \sim \mathcal{D}_1(\varepsilon)$.

Output. Compute θ, the fraction of entries in the table with value 1. Output the t-cropped mean estimation value $f' = 4t \cdot (\theta - 1/2)/\varepsilon + Lap(t/(\varepsilon \cdot m))$.

Figure 2: t-Cropped Mean Estimator

Accuracy for a Close Stream. Observe that if, for every item x, its number of appearances was uniformly and randomly distributed in the range $[2^{\lfloor \log n_x \rfloor}, 2^{\lfloor \log n_x \rfloor + 1}]$, then the problem we described above becomes much less severe. In particular, once we estimate $E \downarrow^N - E \downarrow^{N/2}$ we now know that the expected contribution to this quantity of each item that appeared between $N/2$ and N times is $3N/4$. If there are enough items in this interval then $(E \downarrow^N - E \downarrow^{N/2})/(3N/4)$ is a good approximation to the fraction of items in X that appeared between $N/2$ and N times.

One problem is that the number of appearances of input items is not uniformly or randomly distributed within their respective intervals. The solution is for the streaming algorithm itself to "randomize" the number of times items appear. This is done by choosing a constant $r_x \in_R [1, 2]$ (discretized to precision α/N). For each item in the table maintained by the t-cropped mean algorithm, we count each appearance of the item as r_x appearances (this requires modifying the t-cropped mean algorithm to handle real numbers of appearances, and not only integer numbers). The randomization has the effect of taking each item's number of appearances n_x and randomizing it over the interval $[n_x, 2n_x]$ (discretized up to precision α/N). While the effect is not identical to randomizing n_x within its interval $[2^{\lfloor \log n_x \rfloor}, 2^{\lfloor \log n_x \rfloor + 1}]$, the analysis in the proof of Theorem 4.1 shows that it does provide a similarly flavored guarantee that the "contributions" of items to the cropped means are well distributed and obey strong tail inequalities. For accuracy, note that we have only changed the number of times an item appears by a multiplicative factor of 2, and so the answer we obtain is indeed accurate for a "2-close" stream.

Controlling the Error. Once we have the fraction of items in the "highest" interval (appearing between $N/2$ and N times), we want to approximate the fraction that arrived between $N/2$ and $N/4$ times, between $N/4$ and $N/8$ etc. This is not immediate, because errors might accumulate exponentially fast while computing the approximations for smaller and smaller intervals. We take $t_i = 2^i$ and let the i-th interval be $I_i = [t_i, t_{i+1})$. To estimate the fraction of items appearing in I_i, we again use the cropped mean estimator, together with the previously computed fractions of items appearing in higher intervals. Assume throughout the following informal discussion that each item's number of appearances is randomized within its interval.

Examine the "i-th cropped sum" $E \downarrow^{t_{i+1}} - E \downarrow^{t_i}$. The expected contribution to the i-th cropped sum of each item in interval I_i is $(t_{i+1} - t_i)/2$. There are, however, also contributions from items in higher intervals: each such item contributes exactly $(t_{i+1} - t_i)$ to the i-th cropped sum. To estimate the fraction of items in the i-th interval, we need an estimate z for this contribution of "higher" items. We can then take

$$\left[E \downarrow^{t_{i+1}} - E \downarrow^{t_i} - z\right] / (2/(t_{i+1} - t_i))$$

as an estimate for the fraction of items in the i-th interval. The problem here is that an error of say $\xi \cdot (t_{i+1} - t_i)$ in the estimate z, leads to an error of 2ξ in the estimate of the fraction of items in interval I_i (and this is the type of error we would get from a straightforward use of the fraction of items in higher intervals). Left unchecked, the error might grow exponentially quickly as the algorithm's computation proceeds to lower intervals.

To keep this error under control, we extract an estimate for the "contribution" of items above the i-th interval to the i-th cropped sum (the quantity z above) in a more careful manner. We use v'_{i+1} to denote the (previously computed) estimate for the fraction of items in interval I_{i+1}. Let ξ_{i+1} be the (fractional and signed) error of v'_{i+1}. To estimate z, we estimate separately the contribution of items in interval $i+1$ and the contribution of items *above* interval $i+1$. Both of these quantities are computed using v'_{i+1} (and cropped mean estimations, which have fixed error which does not grow as we move to lower intervals). For the first quantity, the error in computing it is proportional to ξ_{i+1}. For the second quantity, the error is proportional to $-\xi_{i+1}/2$. This means that the two errors cancel out, and the error in estimating z is proportional only to $\xi_{i+1}/2$, which avoids the exponential growth in the error.

More specifically, examine the $(i+1)$-th cropped sum $E \downarrow^{t_{i+2}} - E \downarrow^{t_{i+1}}$. The contribution of the items

above interval $i+1$ to the $(i+1)$-th cropped sum is exactly twice as large as their contribution to the i-th cropped sum. The contribution to the $(i+1)$-th cropped sum of items in interval I_{i+1} is (w.h.p) about $v'_{i+1} \cdot ((t_{i+2} - t_{i+1})/2)$. We can thus compute the $(i+1)$-th cropped sum (using the cropped mean estimator, with fixed error), subtract the expected contribution of items in interval $i+1$, and divide by 2. This gives an estimate for the contribution of items above interval I_{i+1} to the i-th cropped sum. If we have error ξ_{i+1} in our estimate v'_{i+1} to the fraction of items in I_{i+1}, this translates to an error of $-\xi_{i+1}/2 \cdot (t_{i+1} - t_i)$ in the estimate of the contribution of items above interval $i+1$ to the i-th cropped mean (ignoring the error from the cropped mean estimations, which is fixed and never grows). Finally, to compute our approximation z, we also need the contribution of items in I_{i+1} to the i-th cropped sum. This we estimate simply as $v'_{i+1} \cdot (t_{i+1} - t_i)$. An error of ξ_{i+1} in the estimate v'_{i+1}, translates to an error of $\xi_{i+1} \cdot (t_{i+1} - t_i)$ in the estimate of contribution of items in interval I_{i+1} to the i-th cropped sum. Notice now that the two errors have reversed signs, and so the combined error is smaller. We get that the error in the estimate of the contribution of items above interval I_i to the i-th cropped sum behaves like $(\xi_{i+1}/2) \cdot (t_{i+1} - t_i)$. This means that when we estimate the fraction of items in interval I_i by subtracting this contribution and dividing by $(t_{i+1} - t_i)/2$, the error incurred behaves like ξ_{i+1}, and the exponential growth in the error has been avoided.

Summary. In the end we get good approximations to the fraction of t-heavy hitters for $t = 1, 2, 4, 8, \ldots, N/2$. We can take the closest power of 2 to k, say 2^w (assume w.l.o.g. that $2^w < k$), and use the fraction of 2^w-heavy hitters to approximate the fraction of k-heavy hitters: this will be an accurate answer for the "close" stream $\mathcal{S}'$ where every item that appeared between 2^w and k times in $\mathcal{S}$ appears k times in $\mathcal{S}'$. The full algorithm is in Figure 3.

Theorem 4.1. *Assume $\varepsilon \leq 1/2$. When run with parameters $k, N, \varepsilon, \alpha, \beta, \rho$, the k-heavy hitters estimator of Figure 3 guarantees $(\varepsilon \cdot (1 + (\log(N)/\rho)))$-differential pan-privacy. For a fixed input, with probability $1 - \beta$ over the estimator's coins, the output is $\mathrm{poly}(1/\rho, 1/\alpha)$-accurate for a $(1+\rho)$-close stream. The space used is $\mathrm{poly}(\log N, 1/\varepsilon, 1/\alpha, \log(1/\beta), 1/\rho)$.*

The proof is omitted from this extended abstract.

5 Incidence Counting

In this section we obtain an algorithm for estimating the fraction (in X) of t-incidence items, the items that appear exactly t times in the input stream. Here we assume a global upper bound N on the number of times any item can appear in the input stream. Thus the set of *possible incidence counts* is assumed to be $\{0, 1, \ldots, N\}$. Our complexity will be poly-logarithmic in N.

The idea is to hash the possible incidence counts using a hash function h drawn from a family H. Intuitively, this "shrinks" the problem, as the set of possible hash values is much smaller than the set of possible incidents counts. For each possible hash value v, we then keep track (in a pan-private way) of the fraction of items whose incidence count hashes to v. Now, to (over-)estimate the fraction of items that appeared t times, we can simply use the fraction of items whose incidence count hashes to $h(t)$. The hope is that if H is a hash family with low collision probability, then with high probability this is not too high an over-estimate. To avoid an "unlucky" choice of hash function, we can amplify the above idea by choosing independently and at random several functions $h_1, \ldots, h_\ell$, and keeping track of the frequencies of each hash value under each of the hash functions. The final output is the minimum, over $i \in \{1, \ldots, \ell\}$, of the number of items whose frequency hashes, via h_i, to $h_i(t)$.

The idea of hashing to reduce the size of frequency estimation problems appears in the count-min estimator of [5], used to count how many appearances a certain element has. The challenge in our setting, however, is keeping track of the frequencies of hashes of items' *numbers of appearances* (rather than the frequencies of hashes of the items themselves). Moreover, we need to do this computation in a pan-private manner. In particular, this necessitates the use of specific hash families, where the frequencies of hashes of number of appearances can be computed in a pan-private way.

The hash functions we use are modular computations of the form $h(t) = t \bmod p$, where p is a randomly selected prime from the set of the smallest $(4 \log N)/\alpha$ primes and where α is an accuracy parameter. We will use $O(\log(1/\beta))$ hash functions. These functions have small probability of collision, since even if p is drawn from a small set of primes, for an item that appears $n_x \neq t$ times, the probability that $n_x \equiv t \pmod p$ is small. To compute these hash functions in a pan-private way we use the modular incidence counter, described in Section 5.1 below.

A full description of the algorithm appears in Figure 4.

Theorem 5.1. *Assume $\varepsilon \leq 1$. When run with parameters $N, \varepsilon, \alpha, \beta$, the t-incidence estimator of Figure 4 guarantees ε-differential pan-privacy. For any fixed input stream and any desired t, with probability $1-\beta$ over the algorithm's coins, the output is an α-approximation to the fraction (in X) of items that occurred exactly t times in the input stream. The amount of space used is $\mathrm{poly}(\log |N|, 1/\varepsilon, 1/\alpha, \log(1/\beta))$.*

k-**Heavy Hitters Estimator** $(k, N, \varepsilon, \alpha, \beta, \rho)$

Init.

1. Sample a random set M of $m = \text{poly}(\log N, 1/\varepsilon, 1/\alpha, \log(1/\beta), \rho)$ elements (representatives) in X.
2. Assume w.l.o.g. that N is an integer power of $(1+\rho)$. Let $\ell = O(\log_{1+\rho}(N))$, and let: $t_0 = 1, t_1 = (1+\rho), t_2 = (1+\rho)^2, t_3 = (1+\rho)^3, \ldots, t_{\ell-1} = N/(1+\rho), t_\ell = N$. Define $t_{\ell+1}$ to be equal to t_ℓ: $t_{\ell+1} = t_\ell = N$. Take I_i, the i-th interval, to be $I_i = (t_i, t_{i+1}]$.
3. For each t_i, create a table of size m for estimating the t_i-cropped mean with real multiplicities of appearances up to precision α/N. Initialize these tables as done for the t-cropped-mean algorithm.
4. For each $x \in M$, choose $r_x \in_R \{1, 1+\alpha/N, 1+2\alpha/N, 1+3\alpha/N, \ldots, 1+\rho\}$.

Processing. Each time a value $x \in M$ appears in the data stream, update all of the cropped mean counters as if x appeared r_x times in the input stream.

Output.

1. Compute for each $i \in [\ell]$ the (approximate) t_i-cropped mean e'_i.
2. Set $v_\ell \leftarrow 0$.
3. For $i \leftarrow \ell-1, \ell-2, \ldots, 1, 0$, estimate the fraction v'_i of items in interval I_i:
   (a) Estimate the contribution, beyond t_i, of the t_i-heavy-hitters to the t_{i+1}-cropped mean:
   $$e'_{\geq i} \leftarrow e'_{i+1} - e'_i$$
   (b) Estimate the contribution, beyond t_{i+1}, of the t_{i+2}-heavy-hitters to the t_{i+2}-cropped mean:
   $$u_i = (e'_{i+2} - e'_{i+1}) - v'_{i+1} \cdot \frac{t_{i+2} - t_{i+1}}{2}$$
   (c) Estimate the contribution, beyond t_i, of the t_{i+1}-heavy-hitters to the t_{i+1}-cropped mean:
   $$e'_{>i} \leftarrow v'_{i+1} \cdot (t_{i+1} - t_i) + (\frac{t_{i+1} - t_i}{t_{i+2} - t_{i+1}}) \cdot u_i$$
   (d) Estimate the contribution, beyond t_i, of the items in interval I_i to the t_{i+1}-cropped mean:
   $$e'_{=i} \leftarrow e'_{\geq i} - e'_{>i}$$
   (e) Compute the result as the approximate fraction of items in interval I_i:
   $$v'_i \leftarrow \frac{e'_{=i}}{1/2 \cdot (t_{i+1} - t_i)}$$
4. Take $i = \lceil \log_{1+\rho}(k) \rceil$, output $v'_i + Lap(1/(m \cdot \varepsilon))$ as the estimate of the fraction of k-heavy-hitters.

Figure 3: k-Heavy Hitters Estimator $(k, N, \varepsilon, \alpha, \beta, \rho)$

t-**Incidence Estimator**$(N, \varepsilon, \alpha, \beta)$

Init. Let $R = \{1, 2, \ldots, r\}$ be the smallest range of integers containing at least $4 \log N/\alpha$ distinct prime numbers. Choose at random $\ell = O(\log(1/\beta))$ distinct primes $p_0, p_1, \ldots p_{\ell-1}$ from the range R. Run the initialization for the modular incidence counter of Figure 5 for each of these ℓ primes. The i-th incidence counter is initialized with parameters $(p_i, \varepsilon/\ell, \alpha/2, \beta/2)$.

Processing. When a value $x \in M$ appears in the data stream update each of the ℓ modular incidence counters with appearance of the new item x.

Output. For any desired t, estimate the fraction of t-incidence items as follows. For each $i \in [\ell]$ use the i-th modular incidence counter to estimate the fraction f'_i of items that appear $t \pmod{p_i}$ times. Output the (noisy) minimum of these fractions: $\min_{i \in [l]} f'_i$.

Figure 4: Estimating the Fraction of Items Appearing Exactly t Times

The proof is omitted from this extended abstract.

5.1 Modular Incidence Counting

In this section we describe a private streaming algorithm for computing incidence counts modulo an integer k. The setting is of an input data stream of numbers from some universe X. For $i \in \{0, 1, \ldots, k-1\}$ we compute the (approximate) fraction of numbers in the universe X that appear i times $\pmod{k}$. This is done while guaranteeing pan-privacy. This algorithm is a key component in the pan-private incidence counter as described above.

We will again maintain, for a randomly chosen set $M \subset X$, a counter, modulo k, of how many times each element in M appears. In order to preserve the statistics about the frequency of modular incidence counts, we will initialize from a distribution, where the probability of noise value i decreases at a slow exponential rate as i grows from 0 to $k-1$. For pan-privacy, the probabilities of noise values 0 and $k-1$ will differ by only an e^{ε} multiplicative factor.

The similarity of the probabilities of the initial counter values guarantees user-level pan-privacy. The concern now is for accuracy: does the fact that the near-uniform distribution of the initial counter values permit gathering meaningful statistics about the modular counts? To see that recovering statistical information is possible, note that, for example, a counter c_x shows $0 \bmod k$ if any of the following holds: the noise was initialized to 0 and the item showed up a multiple of k times, or the noise was initialized to 1 and the item showed up a number congruent to $k-1 \bmod k$ times, and so on. Using the known distribution on the initialization values as an approximation to the empirical distribution, we can solve a linear program with k equations in k unknowns to approximate the modular incidences.

Theorem 5.2. *Assume $\varepsilon/(k-1) \in [0, 1]$. The modular incidence counter of Figure 5 guarantees 2ε differential pan-privacy (for a single intrusion). For any fixed input stream, with probability $1-\beta$ over the algorithm's coins, for every $i \in [k]$ the counter's output is α-close to the correct fraction of elements in X that occur i times $\pmod{k}$. The amount of space used is $\mathrm{poly}(k, \log|X|, 1/\varepsilon, 1/\alpha, \log(1/\beta))$.*

The proof is omitted from this extended abstract.

6 Lower Bounds

In this section we examine what happens when the adversary can have more than one view of the internal state. In an *unannounced intrusion* the view is surreptitions – the algorithm does not know it has taken place. In an *announced* intrusion the algorithm learns of the intrusion when – but not before – it takes place. This permits the algorithm to inject fresh randomness into the state following the intrusion, and, more generally, to modify its state with the goal of protecting privacy against a subsequent intrusion by the same adversary.

Recall that a density estimator is run on a sequence of elements of X, and must produce (an approximation to) the number of distinct elements in the sequence. We will prove negative results for density estimation in both these settings. The negative result for the case of a few unannounced intrusions applies to *finite state* algorithms.

6.1 Two Unannounced Intrusions

We first show that our density algorithm is maximally tolerant of unannounced intrusions: no pan-private *finite state* algorithm can approximate density in the presence of two unannounced intrusions. Indeed, such an algorithm cannot distinguish between very long streams consisting of a single item and a certain class of streams containing arbitrarily many instances of every item in the universe X.

Theorem 6.1. *For all $\varepsilon \geq 0$, no ε-differentially pan-private finite state algorithm for approximating the stream density can tolerate even two unannounced intrusions.*

Mod-k Incidence Counter$(k, \varepsilon, \alpha, \beta)$

Init. Choose at random a set M of $m = \text{poly}(k, \log|X|, 1/\varepsilon, 1/\alpha, \log(1/\beta))$ elements (representatives) in X. Create a table of size m with a single entry for each item in M. For every entry $x \in M$, generate a random initial integer noise value in $\{0, 1, \ldots, k-1\}$, where noise value i is chosen with probability proportional to $e^{-\varepsilon \cdot i/(k-1)}$.

Processing. When a value $x \in M$ appears in the data stream update x's entry in the table: increase it by 1 $\pmod k$.

Output. To compute the output, for every $i \in [k]$ compute the fraction s_i of table entries with value i. Let $\vec{s}$ be the vector of these fractions (summing up to 1). Let $\vec{q}$ be the vector obtained by adding independent noise drawn from $Lap(1/(\varepsilon \cdot m))$ to each entry of $\vec{s}$. Take N to be the $k \times k$ noise matrix such that

$$N_{i,j} = \frac{e^{(-\varepsilon/(k-1)) \cdot (i+j \pmod k)}}{\sum_{j=0}^{k-1} e^{-\varepsilon j/(k-1)}}$$

Solve a linear program to obtain a k-entry frequency vector $\vec{f'}$. The linear program is specified by the equality $\sum_{i=0}^{k-1} \vec{f_i'} = 1$ and the inequalities:

$$\vec{q} - (\alpha/2, \ldots, \alpha/2) \leq N \times \vec{f'} \leq \vec{q} + (\alpha/2, \ldots, \alpha/2)$$

Output the obtained $\vec{f'}$ as the vector of modular fractional incidences (i.e. entry i in the vector is the fraction of items that appeared i times $\pmod k$).

Figure 5: Mod-k Incidence Counter

Proof. We prove that for any $0 \leq \xi \leq 1$ there exists a fixed $K = K_\xi$ and two classes of streams, whose definition depends on K, such that the density of the first class is $1/|X|$ and the density of the second class is 1, and such that the statistical distance between the behavior of the algorithm on the first and second classes is at most ξ. We will use the term *mixing time* for an aperiodic Markov chain to mean the maximum, over all possible starting states s for the chain, of the minimum time t such that the statistical distance between the probability distribution of the chain when started at s and run for t steps, and the stationary distribution of the chain, is at most $\xi/4$.

Assume for the sake of contradiction that an algorithm exists. and let Σ denote the set of states. For any symbol (user name) $a \in X$, let M^a denote the state transition matrix of the algorithm on input a, so for all $i, j \in \Sigma$, M^a_{ij} is the probability of entering state j from state i when the input symbol read is a.

Consider an arbitrary atomic read-and-update event. Since the adversary can intrude immediately before and immediately after this event, and since we require differential privacy, if a given state transition can occur on some symbol, it can occur on all symbols.

Formally, For every pair $i, j \in \Sigma$, and every pair of symbols $a, b \in X$, we have

$$M^a_{ij} = 0 \Leftrightarrow M^b_{ij} = 0. \tag{2}$$

(We can make a stronger statement, involving ε, but we don't use this fact in the proof.)

The transition matrices M^a, for $a \in X$, define Markov chains on a common state space. A group of states of a Markov chain forms an *ergodic set* if it is possible to get from each state in the ergodic set to each other state, and once within the ergodic set, the process does not leave it. A state is *transient* if it is not a member of an ergodic set. Thus for each a, we can partition the states of M^a into ergodic sets, and transient states. From Equation 2 we know that, for all $a, b \in X$, the set of ergodic sets is the same. Let $C_1, C_2, \ldots, C_m$ denote these ergodic sets. Similarly, we know that the set of transient states is the same.

For now, assume that each C_j, $1 \leq j \leq m$, is aperiodic, so all the chains are aperiodic and it makes sense to speak of their mixing times. (1) Fix an arbitrary $a \in X$ and run the algorithm on input a^K. Since K is at least the mixing time of M^a, and since the transient states have no support in the stationary distribution, with probability at least $1 - \xi/4$ the algorithm will be trapped in some C_j.

(2) For all $1 \leq j \leq m$, once a prefix of the data stream brings the algorithm to a state in C_j, an algorithm cannot distinguish the following two types of suffices: a^K for an arbitrary $a \in X$, where K is greater than the mixing time of the chains, and $X^* a^K$. In particular, it cannot distinguish between a suffix that has some large number n of distinct elements of X, followed by just a single name $a \in X$ many times, and a long string consisting of no symbol other than a. This is because the statistical distance between the resulting distributions on the states is at most $2\xi/4 = \xi/2$.

Combining (1) and (2), it follows that the statistical distance between the resulting distributions when we run the algorithm on the two strings (i) $a^K a^K$ and (ii) $a^K X^n a^K$ is at most $\xi/4 \cdot 1 + 2\xi/4 = 3\xi/4$.

We now argue that periodicity does not affect this ar-

gument.

Fix an input symbol, say, $a \in X$ and look at the chain M^a. Consider a particular ergodic set C_j. Assume it has a period d. Letting P denote the transition matrix of C_j, the matrix P^d is the transition matrix of a Markov chain with d separate ergodic sets $A^a_{j1}, \ldots, A^a_{jd}$, each of which is aperiodic. As before, by Equation 2, for all $a, b \in X$, all $1 \leq j \leq m$, and all $1 \leq k \leq d$, we have $A^a_{jk} = A^b_{jk}$.

Now let K be the maximum, over all $a \in X$, $1 \leq j \leq m$, $1 \leq k \leq d$, of the mixing time of A^a_{jk}. Also, choose K sufficiently large so that with probability at least $1 - \xi/4$, the algorithm will reach an ergodic set when run on prefix a^K, for any $a \in \Sigma$.

The argument now is as before. For an arbitrary symbol a, run the algorithm on a^K, after which, with high probability, it has been driven into one of the ergodic sets of M^a, say C_j. The algorithm cannot distinguish between the following two suffixes: a^{Kd} and $(X^*)^d a^{Kd}$. □

We remark that the sets of executions constructed in the proof of Theorem 6.1 do not actually suffer from intrusions. It is merely the threat of the intrusions that forces the algorithm to satisfy Equation 2. This threat is present even if the second intrusion is announced. This is remeniscent of the proof of impossibility of tolerating even one uannounced process failure in an asynchronous distributed consensus protocol [20]. It also yields the following immediate corollary.

Corollary 6.2. *Theorem 6.1 holds also for the case of a single unannounced intrusion followed by an announced intrusion.*

6.2 Continual (Announced) Intrusions

Here we show that in the presence of continual intrusions it is impossible to estimate the density in a pan-private manner with error $o(n)$. In other words, in the presence of continual intrusions any density estimator must have asymptotically trivial error.

Theorem 6.3. *Any density estimator that is ε-pan-private in the presence of continual intrusions must have error $\Omega(n)$ with probability at least $1/2 - \sqrt{2\varepsilon}$.*

The proof is omitted from this extended abstract.

7 Streaming vs. Sketching

In this section we show that there are functions or relations that can be computed with reasonable accuracy in the streaming model with pan privacy, but cannot be computed in the sketching model with any amount of accuracy. In the sketching model each party acts on its own and publishes information based on its own input. The value of the function is then computed from this published information. The advantage of this model is that the participants need not trust anyone. In comparison, in the streaming model the participants send their (sensitive) inputs to the algorithm, and these inputs are used to update the algorithm's (pan-private) state.

To separate the models we assume here that the adversary sees the output of just one party in the sketching case (given the independent moves by each party, this is w.l.o.g). In the streaming model we allow continual intrusions, that is the adversary sees all the state at any point in time (this only strengthens the separation).

The separating function (strictly speaking this is a relation) is not symmetric (an interesting question is whether the two models are equivalent for symmetric functions). The participant are divided into two groups, called the y part and the x part. The input consists of a collection of y's which are binary and X's which are binary vectors of length m (that is participants that belong to the y part have a single bit as their input and those that belong to the x part have an m bit vector as input). The y's determine a binary vector V of length m, by means of a majority; they are partitioned into m sets and each set, which is of size k, is responsible for one entry in V: a majority of the y's in the set determines the corresponding bit in V. Let $\langle V, X \rangle$ denote the inner product over $GF[2]$. The output of the function is the vector V together with $\sum_i \langle V, X^i \rangle$.

In other words the input is

$$(y^1_1, y^2_1, ...y^k_1, y^1_2, y^2_2, ...y^k_2 ...y^1_m, y^2_m ...y^k_m, X^1, X^2, ..., X^n)$$

and the output is (V, Z) where V should be an *approximate* majority of the y's (that is if $\sum_{j=1}^k y^j_i$ is at most $k/3$ then V_i should be 0, if it is at least $2/3k$ it should be 1 and it can be any value otherwise) and Z should be an approximation to $\sum_{i=1}^n \langle V, X^i \rangle$. We require that $m \gg \log n$ and $k \gg \log m$. Call this the *inner product counter*.

Pan Private Computation in the Streaming Model. To compute this function in the streaming model, we first process the y part to determine V. This is done by each y participant outputting a noisy version for its input y_j, as in randomized response. The state simply counts how many times each value was output. This allows computing with very high probability a correct value for V (provided that k is at least logarithmic in m). When we get to the x part, at the jth step the state consists of the approximation of V (computed from the y values) and the number of times the participants whose inputs were $X^1, \ldots, X^{j-1}$ had inner-product '1' with V. At the jth step a noisy version of $\langle V, X^j \rangle$ is output and the count is added to the state. At the end of the computation, an approximation to $\sum_{i=1}^n \langle V, X^i \rangle$ is computed and this together with V is the output. Note that this process is actually pan private with continual intrusion. In other words there is no secret state at any point in time.

Impossibility of Pan Private Computation in the Sketching Model. We claim that this function cannot be computed in the sketching model with pan privacy against a single intrusion. The intuition is that each participant in the x part has to guess the correct V. Otherwise he will be outputting enough information to approximate $\langle V, X^i \rangle$ for many values of V; this, in turn, is sufficient information for reconstructing X^i, in the style of Goldreich and Levin [21].

The full proof is omitted from this extended abstract.

References

[1] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevy and Tal Rabin, editors, *First Theory of Cryptography Conference (TCC)*, volume 3876, pages 265–284. Springer-Verlag, 2006.

[2] C. Dwork. Differential privacy. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP)(2)*, pages 1–12, 2006.

[3] S. L. Warner. Randomized response: a survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60:63–69, 1965.

[4] N. Mishra and M. Sandler. Privacy via pseudorandom sketches. In *25th Annual ACM Symposium on Principles of Database Systems (PODS)*, 2006.

[5] Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[6] S. Muthukrishnan. Data stream algorithms (notes from barbados 09 complexity theory meeting), 2009. Available at http://algo.research.googlepages.com/datastreamalgorithms.

[7] L. Bottou and Y. Le Cun. On-line learning for very large datasets. *Applied Stochastic Models in Business and Industry*, 21:137–151, 2005.

[8] M. Datar and S. Muthukrishnan. Estimating rarity and similarity over data stream windows. In *ESA*, pages 323–334, 2002.

[9] P. Indyk. Approximate nearest neighbor under edit distance via product metrics. In *SODA*, pages 646–650, 2004.

[10] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS*, pages 211–222, 2003.

[11] Joe Kilian, André Madeira, Martin J. Strauss, and Xuan Zheng. Fast private norm estimation and heavy hitters. In *TCC*, pages 176–193, 2008.

[12] André Madeira and S. Muthu Muthukrishnan. Functionally private approximation for negligibly-biased estimators. In *FSTTCS (to appear)*, 2009.

[13] D. Feldman, A. Fiat, H. Kaplan, and K. Nissim. Private coresets. In *STOC*, 2009.

[14] Daniele Micciancio. Oblivious data structures: Applications to cryptography. In *STOC*, pages 456–464, 1997.

[15] Moni Naor and Vanessa Teague. Anti-presistence: history independent data structures. In *STOC*, pages 492–501, 2001.

[16] Boaz Barak and Shai Halevi. A model and architecture for pseudo-random generation with applications to /dev/random. In *ACM Conference on Computer and Communications Security*, pages 203–212, 2005.

[17] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, November 1984.

[18] Andrew C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

[19] S. Chien, K. Ligett, and A. McGregor. Space-efficient estimation of robust statistics and distribution testing. Manuscript in submission, 2009.

[20] Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.

[21] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 25–32, 1989.