Adversarially Robust Streaming Algorithms via Differential Privacy

AVINATAN HASSIDIM, Bar-Ilan University and Google
HAIM KAPLAN and YISHAY MANSOUR, Tel Aviv University and Google
YOSSI MATIAS, Google
URI STEMMER, Tel Aviv University and Google

A streaming algorithm is said to be *adversarially robust* if its accuracy guarantees are maintained even when the data stream is chosen maliciously, by an *adaptive adversary*. We establish a connection between adversarial robustness of streaming algorithms and the notion of *differential privacy*. This connection allows us to design new adversarially robust streaming algorithms that outperform the current state-of-the-art constructions for many interesting regimes of parameters.

CCS Concepts: • **Theory of computation → Streaming models;**

Additional Key Words and Phrases: Streaming, robustness, differential privacy

ACM Reference format:
Avinatan Hassidim, Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. 2022. Adversarially Robust Streaming Algorithms via Differential Privacy. *J. ACM* 69, 6, Article 42 (November 2022), 14 pages.
https://doi.org/10.1145/3556972

1 INTRODUCTION

The field of *streaming algorithms* was formalized by Alon et al. [3] and has generated a large body of work that intersects many other fields in computer science, such as theory, databases, networking, and natural language processing. Consider a scenario in which data items are being generated one by one, e.g., IP traffic monitoring or web searches. Generally speaking, streaming algorithms aim to process such data streams while using only a limited amount of memory, significantly smaller than what is needed to store the entire data stream.[1] Typical streaming problems include estimating

[1]We remark, however, that streaming algorithms are also useful in the offline world, e.g., to process a large unstructured database that is located on an external storage.

Haim Kaplan was supported in part by the Israel Science Foundation (grant 1595/19) and by the Blavatnik family foundation. Yishay Mansour was supported in part by funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant 882396), by the Israel Science Foundation (grant 993/17), Tel Aviv University Center for AI and Data Science (TAD), and the Yandex Initiative for Machine Learning at Tel Aviv University. Uri Stemmer was supported in part by the Israel Science Foundation (grant 1871/19) and by the Blavatnik family foundation.
Authors' addresses: A. Hassidim, H. Kaplan, Y. Mansour, Y. Matias, and U. Stemmer, Google Tel Aviv, Electra Tower, Yigal Alon 98 Tel Aviv, 6789141 Israel; emails: avinatan@google.com, haimk@tau.ac.il, mansour.yishay@gmail.com, yossi@google.com, u@uri.co.il.

0004-5411/2022/11-ART42 $15.00
https://doi.org/10.1145/3556972

frequency moments, counting the number of distinct elements in the stream, identifying heavy-hitters in the stream, estimating the median of the stream, and much more [5, 15–18, 23, 24, 30, 33, 37, 40, 41].

Usually, streaming algorithms can be queried many times throughout the execution. The reason is that (usually) the space requirement of streaming algorithms scales as $\log(1/\delta)$, where δ is the failure probability of the algorithm. By a union bound, this means that to guarantee accuracy for m queries (with probability $1-\delta$) the space only scales proportionally to $\log(m/\delta)$, so we can tolerate a substantial number of queries without blowing up the space. However, for this argument to go through, we need to assume that the entire stream is *fixed* in advanced (and is just given to us one item at a time), or at least that the choice of the items in the stream is *independent* of the internal state (and coin tosses) of our algorithm. This setting is sometimes referred to as the *oblivious* setting. Most of the work on streaming algorithms is focused on the oblivious setting.

Now suppose that the items in the stream, as well as the queries issued to the algorithm, are chosen by an *adaptive (stateful) adversary*. Specifically, every item in the stream (and each of the queries) is chosen by the adversary as a function of the previous items in the stream, the previous queries, and the previous answers given by our streaming algorithm. As a result, the items in the stream are *no longer independent* of the internal state of our algorithm. Oblivious streaming algorithms fail to provide meaningful utility guarantees in such a situation. In this work, we aim to design *adversarially robust streaming algorithms* that maintain (provable) accuracy against such adaptive adversaries while, of course, keeping the memory and runtime requirements to a minimum. We stress that such dependencies between the items in the stream and the internal state of the algorithm may occur unintentionally (even when there is no "adversary"). For example, consider a large system in which a streaming algorithm is used to analyze data coming from one part of the system while answering queries generated by another part of the system, but these (supposedly) different parts of the system are connected via a feedback loop. In such a case, it is no longer true that the items in the stream are generated independently of the previous answers, and most of the existing streaming algorithms would fail to provide meaningful utility guarantees.

Recall that (typically) in the *oblivious setting*, the memory requirement only grows logarithmically with the number m of queries that we want to support. For the *adaptive setting*, one can easily show that a memory blowup of $\tilde{O}(m)$ suffices. This can be achieved, e.g., by running m independent copies of the algorithm (where we feed the input stream to each of the copies) and using each copy to answer at most one query. Can we do better?

This question has motivated a recent line of work that is focused on constructing *adversarially robust streaming algorithms* [1, 2, 10, 11, 25, 26, 29, 39]. The formal model we consider was recently put forward by Ben-Eliezer et al. [10], who presented adversarially robust streaming algorithms for many problems in the *insertion-only model*, i.e., when the stream contains only *positive* updates. Moreover, their results extend to *turnstile streams* (where both *positive* and *negative* updates are allowed), provided that the number of negative updates is small. The question remained largely open for the general turnstile model where there might be a large number of negative updates.

1.1 Existing Results

We now give an informal overview of the techniques of Ben-Eliezer et al. [10]. This intuitive overview is generally oversimplified and hides many of the difficulties that arise in the actual analysis. See their work [10] for the formal details and for additional results.

Consider a stream of updates $u_1, \ldots, u_m$, where every update $u_i \in X$ comes from a domain X of size $|X| = n$. For $i \in [m]$, we write $\vec{u}_i = (u_1, \ldots, u_i)$ to denote the first i elements of the stream. Let $g : X^* \to \mathbb{R}$ be a function, e.g., g might count the number of distinct elements in the stream.

At every timestep i, after obtaining the next element in the stream u_i, our goal is to output an approximation for $g(\vec{u}_i)$.

To illustrate the results of Ben-Eliezer et al. [10], let us consider the *distinct elements* problem (in the *insertion only* model, where there are no deletions in the stream). Specifically, after every update u_i we need to output an estimate of distinct elements in the stream, i.e., $g(\vec{u}_i) = |\{u_j : j \in [i]\}|$. Observe that this quantity is monotonically increasing. Furthermore, since we are aiming for a multiplicative $(1 \pm \alpha)$ error, even though the stream is large (of length m), the number of times we actually need to modify the estimates we release is quite small, roughly $\frac{1}{\alpha} \log m$ times. (We remark that this is not true if the stream might contain *deletions*, a.k.a. the *turnstile model*.) Informally, the idea of Ben-Eliezer et al. [10] is to run several independent sketches in parallel, and to use each sketch to release answers over a part of the stream during which the estimate remains constant. In more detail, the generic transformation of Ben-Eliezer et al. [10] (applicable not only to the distinct elements problem) is based on the following definition.

Definition 1.1 (Flip Number [10]). Given a function g, the (α, m)*-flip number* of g, denoted as $\lambda_{\alpha,m}(g)$, is the maximal number of times that the value of g can change (increase or decrease) by a factor of $(1 + \alpha)$ during a stream of length m.

The generic construction of Ben-Eliezer et al. [10] for a function g is as follows:

(1) Instantiate $\lambda \geq \lambda_{\alpha,m}(g)$ independent copies of an oblivious streaming algorithm for the function g, and set $j = 1$.
(2) When the next update u_i arrives:
   (a) Feed u_i to *all* of the λ copies.
   (b) Release an estimate using the jth copy (rounded to the nearest power of $(1 + \alpha)$). If this estimate is different from the previous estimate, then set $j \leftarrow j + 1$.

Ben-Eliezer et al. [10] showed that this can be used to transform an oblivious streaming algorithm for g into an adversarially robust streaming algorithm for g. In addition, the overhead in terms of memory is only $\lambda_{\alpha,m}(g)$, which is typically small in the insertion-only model (typically $\lambda_{\alpha,m}(g) \lesssim \frac{1}{\alpha} \log m$). Moreover, Ben-Eliezer et al. [10] showed that their techniques extend to the *turnstile model* (when the stream might contain deletions), provided that the number of negative updates is small (and so $\lambda_{\alpha,m}(g)$ remains small).

Theorem 1.2 ([10], Informal). *Fix any function g and let $\mathcal{A}$ be an oblivious streaming algorithm for g that for any $\alpha, \delta > 0$ uses space $L(\alpha, \delta)$ and guarantees accuracy α with success probability $1 - \delta$ for streams of length m. Then there exists an adversarially robust streaming algorithm for g that guarantees accuracy α with success probability $1 - \delta$ for streams of length m using space*

$$O\left(L\left(\frac{\alpha}{10}, \delta\right) \cdot \lambda_{\frac{\alpha}{10},m}(g)\right).$$

1.2 Our Results

We establish a connection between adversarial robustness of streaming algorithms and *differential privacy*, a model to provably guarantee privacy protection when analyzing data. Consider a database containing (sensitive) information pertaining to individuals. An algorithm operating on such a database is said to be *differentially private* if its outcome does not reveal information that is specific to any individual in the database. More formally, differential privacy requires that no individual's data has a significant effect on the distribution of the output. Intuitively, this guarantees that whatever is learned about an individual could also be learned with her data arbitrarily modified (or without her data). Formally, we have the following definition.

Definition 1.3 ([20]). Let $\mathcal{A}$ be a randomized algorithm that operates on databases. Algorithm $\mathcal{A}$ is (ε, δ)-*differentially private* if for any two databases S, S' that differ on one row, and any subset of outcomes T, we have

$$\Pr[\mathcal{A}(S) \in T] \leq e^{\varepsilon} \cdot \Pr[\mathcal{A}(S') \in T] + \delta.$$

Our main conceptual contribution is to show that the notion of differential privacy can be used *as a tool* to construct new adversarially robust streaming algorithms. In a nutshell, the idea is to protect the *internal state* of the algorithm using *differential privacy*. Loosely speaking, this limits (in a precise way) the dependency between the internal state of the algorithm and the choice for the items in the stream, and allows us to analyze the utility guarantees of the algorithm even in the adaptive setting. Notice that differential privacy is *not* used here to protect the privacy of the data items in the stream. Rather, differential privacy is used here to protect the internal randomness of the algorithm.

For many problems of interest, even in the general turnstile model (with deletions), this technique allows us to obtain adversarially robust streaming algorithms with sublinear space. To the best of our knowledge, our technique is the first to provide meaningful results for the general turnstile model. In addition, for interesting regimes of parameters, our algorithm outperforms the results of Ben-Eliezer et al. [10] also for the insertion-only model (strictly speaking, our results for the insertion-only model are incomparable with their work [10]).

We obtain the following theorem.

Theorem 1.4. *Fix any function g and fix $\alpha, \delta > 0$. Let $\mathcal{A}$ be an oblivious streaming algorithm for g that uses space $L(\frac{\alpha}{10}, \frac{1}{10})$ and guarantees accuracy $\frac{\alpha}{10}$ with success probability $\frac{9}{10}$ for streams of length m. Then there exists an adversarially robust streaming algorithm for g that guarantees accuracy α with success probability $1 - \delta$ for streams of length m using space*

$$O\left(L\left(\frac{\alpha}{10}, \frac{1}{10}\right) \cdot \sqrt{\lambda_{\frac{\alpha}{10}, m}(g) \cdot \log\left(\frac{1}{\delta}\right)} \cdot \log\left(\frac{m}{\alpha\delta}\right)\right).$$

Compared to Ben-Eliezer et al. [10], our space bound grows only as $\sqrt{\lambda}$ instead of linearly in λ. This means that in the general turnstile model, when λ can be large, we obtain a significant improvement at the cost of additional logarithmic factors. In addition, as λ typically scales at least linearly with $1/\alpha$, we obtain improved bounds even for the insertion-only model in terms of the dependency of the memory in $1/\alpha$ (again, at the expense of additional logarithmic factors).

1.3 Follow-Up Work

Following our work, Woodruff and Zhou [46] presented a different (non black-box) transformation that (under additional assumptions) results in improved space requirements for insertion-only streams, and incomparable space requirements for turnstile streams. Later, Attias et al. [4] combined the results of Woodruff and Zhou [46] with our techniques to get improved results for turnstile streams. These results are incomparable with our results, as they require additional assumptions and do not result in black-box reductions.

Ben-Eliezer et al. [9] studied the case where the flip-number λ is unbounded and can be as big as the length of the stream m. Building on our results, they presented robust algorithms for this setting with improved space complexity for estimating frequencies moments. In particular, for estimating the number of distinct elements (F_0) and for estimating the squared Euclidean norm of the frequencies vector (F_2), they obtained space complexity $\tilde{O}(m^{1/3})$ and $\tilde{O}(m^{2/5})$, respectively.

Our results were shown to be optimal by Kaplan et al. [35]. Specifically, they presented a (non-natural) streaming problem for which the space complexity resulting from our transformation is tight (up to lower-order terms).

1.4 Other Related Results

Over the past few years, differential privacy has proven itself to be an important *algorithmic* notion (even when data privacy is not of concern) and has found itself useful in many other fields, such as machine learning, mechanism design, secure computation, probability theory, secure storage, and more [6, 7, 19, 28, 36, 38, 42–44]. In particular, our results utilize a connection between *differential privacy* and *generalization*, which was first discovered by Dwork et al. [19] in the context of *adaptive data analysis*.

2 PRELIMINARIES

A stream of length m over a domain X (of size $|X| = n$) consists of a sequence of updates $u_1, \ldots, u_m$, where $u_i \in X$. For $i \in [m]$, we write $\vec{u}_i = (u_1, \ldots, u_i)$ to denote the first i elements of the stream. Let $g : X^* \to \mathbb{R}$ be a function, e.g., g might count the number of distinct elements in the stream. At every timestep i, after obtaining the next element in the stream u_i, our goal is to output an approximation for $g(\vec{u}_i)$. We assume throughout the article that $\log(m) = \Theta(\log n)$ and that g is polynomially bounded in n.

2.1 Streaming Against an Adaptive Adversary

The adversarial streaming model, in various forms, was considered in several works [1, 2, 10, 11, 25, 26, 29, 39]. We give here the formulation presented by Ben-Eliezer et al. [10]. The adversarial setting is modeled by a two-player game between a (randomized) `StreamingAlgorithm` and an `Adversary`. At the beginning, we fix a function g. Then the game proceeds in rounds, where in the ith round:

(1) The `Adversary` chooses an update u_i for the stream, which can depend, in particular, on all previous stream updates and outputs of the `StreamingAlgorithm`.
(2) The `StreamingAlgorithm` processes the new update u_i and outputs its current response z_i.

The goal of the `Adversary` is to make the `StreamingAlgorithm` output an incorrect response z_i at some point i in the stream. For example, in the distinct elements problem, the adversary's goal is that at some step i, the estimate z_i will fail to be a $(1+\alpha)$-approximation of the true current number of distinct elements.

We remark that our techniques extend to a model in which the `StreamingAlgorithm` does not need to release an output on every timestep, only at *query times* (which are chosen adaptively by the adversary), in exchange for lower space requirements as a function of the allowed number of queries. See Appendix A for more details.

2.2 Preliminaries from Differential Privacy

The Laplace mechanism. The most basic constructions of differentially private algorithms are via the Laplace mechanism as follows.

Definition 2.1 (The Laplace Distribution). A random variable has probability distribution $\mathrm{Lap}(b)$ if its probability density function is $f(x) = \frac{1}{2b}\exp(-\frac{|x|}{b})$, where $x \in \mathbb{R}$.

Definition 2.2 (Sensitivity). A function $f : X^* \to \mathbb{R}$ has *sensitivity* ℓ if for every two databases $S, S' \in X^*$ that differ in one row, it holds that $|f(S) - f(S')| \leq \ell$.

Theorem 2.3 (The Laplace Mechanism [20]). *Let $f : X^* \to \mathbb{R}$ be a function of sensitivity ℓ. The mechanism that on input $S \in X^*$ returns $f(S) + \mathrm{Lap}(\frac{\ell}{\varepsilon})$ preserves $(\varepsilon, 0)$-differential privacy.*

Example 2.4. Consider a database S containing the medical records of n individuals. Suppose that we are interested in privately estimating the number of individuals with diabetes, and let

$f(S)$ denote the function that, given S, returns the number of records in S that correspond to individuals with diabetes. Observe that the sensitivity of f is 1, since modifying a record in the data can change the number of individuals with diabetes by at most 1. Therefore, Theorem 2.3 states that we can privately estimate $f(S)$ by adding noise sampled from $\mathrm{Lap}(\frac{1}{\varepsilon})$. Observe that the noise magnitude is independent of the database size. Hence, when the database S is large enough, the noise we add for privacy has only a very small (relative) effect on the result.

The sparse vector technique. Consider a large number of low-sensitivity functions $f_1, f_2, \ldots$ that are given (one by one) to a data curator (holding a database S). Dwork et al. [21] presented a simple (and elegant) tool that can *privately* identify the first index i such that the value of $f_i(S)$ is "large."

Algorithm `AboveThreshold`
Input: Database $S \in X^*$, privacy parameter ε, threshold t, and a stream of sensitivity-1 queries $f_i : X^* \to \mathbb{R}$.
(1) Let $\hat{t} \leftarrow t + \mathrm{Lap}(\frac{2}{\varepsilon})$.
(2) In each round i, when receiving a query f_i, do the following:
  (a) Let $\hat{f}_i \leftarrow f_i(S) + \mathrm{Lap}(\frac{4}{\varepsilon})$.
  (b) If $\hat{f}_i \geq \hat{t}$, then output $\top$ and halt.
  (c) Otherwise, output $\perp$ and proceed to the next iteration.

Notice that the number of possible rounds is unbounded. Nevertheless, this process preserves differential privacy.

Theorem 2.5 ([21, 27]). *Algorithm* `AboveThreshold` *is* $(\varepsilon, 0)$*-differentially private.*

Privately approximating the median of the data. Given a database $S \in X^*$, consider the task of *privately* identifying an *approximate median* of S. Specifically, for an error parameter Γ, we want to identify an element $x \in X$ such that there are at least $|S|/2 - \Gamma$ elements in S that are larger or equal to x, and there are at least $|S|/2 - \Gamma$ elements in S that are smaller or equal to x. The goal is to keep Γ as small as possible, as a function of the privacy parameters ε, δ, the database size $|S|$, and the domain size $|X|$.

There are several advanced constructions in the literature with error that grows very slowly as a function of the domain size (only polynomially with $\log^* |X|$) [8, 13, 14, 34]. In our application, however, the domain size is already small, and hence we can use simpler constructions (where the error grows logarithmically with the domain size). The following theorem can be derived as an immediate application of the exponential mechanism [38].

Theorem 2.6. *There exists an* $(\varepsilon, 0)$*-differentially private algorithm that given a database* $S \in X^*$ *outputs an element* $x \in X$ *such that with probability at least* $1 - \delta$ *there are at least* $|S|/2 - \Gamma$ *elements in* S *that are bigger or equal to* x*, and there are at least* $|S|/2 - \Gamma$ *elements in* S *that are smaller or equal to* x*, where* $\Gamma = O(\frac{1}{\varepsilon} \log(\frac{|X|}{\delta}))$.

Composition of differential privacy. The following theorem allows to argue about the privacy guarantees of an algorithm that accesses its input database using several differentially private mechanisms.

Theorem 2.7 ([22]). *Let* $0 < \varepsilon, \delta' \leq 1$*, and let* $\delta \in [0, 1]$*. A mechanism that permits* k *adaptive interactions with mechanisms that preserve* (ε, δ)*-differential privacy (and does not access the database otherwise) ensures* $(\varepsilon', k\delta + \delta')$*-differential privacy, for* $\varepsilon' = \sqrt{2k \ln(1/\delta')} \cdot \varepsilon + 2k\varepsilon^2$.

Generalization properties of differential privacy. Dwork et al. [19] and Bassily et al. [6] showed that if a predicate h is the result of a differentially private computation on a random sample, then the empirical average of h and its expectation over the underlying distribution are guaranteed to be close.

Theorem 2.8 ([6, 19, 45]). *Let $\varepsilon \in (0, 1/3)$, $\delta \in (0, \varepsilon/4)$, $m \in \mathbb{N}$, and $n \geq \frac{1}{\varepsilon^2}\log(\frac{2\varepsilon m}{\delta})$. Let $\mathcal{A} : X^n \to (2^X)^m$ be an (ε, δ)-differentially private algorithm that operates on a database of size n and outputs m predicates $h_1, \ldots, h_m : X \to \{0, 1\}$. Let $\mathcal{D}$ be a distribution over X, let S be a database containing n i.i.d. elements from $\mathcal{D}$, and let $(h_1, \ldots, h_m) \leftarrow \mathcal{A}(S)$. Then*

$$\Pr_{\substack{S\sim\mathcal{D}^n \\ (h_1,\ldots,h_m)\leftarrow\mathcal{A}(S)}}\left[\max_{1\leq i\leq m}\left|\frac{1}{|S|}\sum_{x\in S}h_i(x) - \mathbb{E}_{x\sim\mathcal{D}}[h_i(x)]\right| \geq 10\varepsilon\right] < \frac{\delta}{\varepsilon}.$$

3 DIFFERENTIAL PRIVACY AS A TOOL FOR ROBUST STREAMING

In this section, we present our main construction—algorithm RobustSketch. Recall that the main challenge when designing adversarially robust streaming algorithms is that the elements in the stream can depend on the internal state of the algorithm. To overcome this challenge, we protect the internal state of algorithm RobustSketch using differential privacy.

Suppose that we have an oblivious streaming algorithm $\mathcal{A}$ for a function g. In our construction, we run k independent copies of $\mathcal{A}$ with independent randomness and feed the input stream to all of the copies. When a query comes, we aggregate the responses from the k copies in a way that protects the internal randomness of each of the copies using differential privacy. In addition, assuming that the *flip number* [10] of the stream is small, we get that the number of times that we need to compute such an aggregated response is small. We use the sparse vector technique (algorithm AboveThreshold) [21] to identify the timesteps in which we need to aggregate the responses of the k copies of $\mathcal{A}$, and the aggregation itself is done using a differentially private algorithm for approximating the *median* of the responses.

In the next lemma, we show that algorithm RobustSketch satisfies differential privacy w.r.t. the internal randomness of the different copies of $\mathcal{A}$. In our case, it suffices to guarantee differential privacy with a constant ε, and we fix $\varepsilon = \frac{1}{100}$.

Lemma 3.1. *Denote $\varepsilon = \frac{1}{100}$, and let $\delta \in (0, 1)$ be a parameter. Algorithm* RobustSketch *satisfies (ε, δ)-differential privacy (w.r.t. the collection of strings R).*

Proof. Each execution of the outer loop consists of applying algorithm AboveThreshold and applying algorithm PrivateMed, each of which satisfies $(\varepsilon_0, 0)$-differential privacy. The lemma now follows from composition theorems for differential privacy (see Theorem 2.7). □

Recall that algorithm RobustSketch might halt before the stream ends. In the following lemma, we show that if $k = \tilde{\Omega}(\sqrt{\lambda})$, then (w.h.p.) all of the answers that RobustSketch returns before it halts are accurate. Afterward, in Lemma 3.3, we show that if $\lambda \geq \lambda_{\alpha/10,m}(g)$, then (w.h.p.) the algorithm does not halt prematurely.

Lemma 3.2. *Let $\mathcal{A}$ be an oblivious streaming algorithm for a function g, which guarantees that with probability at least $9/10$, all of its estimates are accurate to within multiplicative error of $(1 \pm \frac{\alpha}{10})$. Then, with probability at least $1 - \delta$, all of the estimates returned by* RobustSketch *before it halts are accurate to within multiplicative error of $(1 \pm \alpha)$, even when the stream is chosen by an adaptive adversary, provided that*

$$k = \Omega\left(\sqrt{\lambda \cdot \log\left(\frac{1}{\delta}\right)} \cdot \log\left(\frac{m}{\alpha\delta}\right)\right),$$

where m is the length of the stream.

ALGORITHM : `RobustSketch`

Input: Parameters $\alpha, \lambda, \delta, k$, and a collection of k random strings $R = (r_1, \ldots, r_k) \in (\{0,1\}^*)^k$.
Algorithm used: An oblivious streaming algorithm $\mathcal{A}$ for a functionality g that guarantees that with probability at least 9/10, all of its estimates are accurate to within a multiplicative error of $(1 \pm \frac{\alpha}{10})$.

(1) Initialize k independent instances $\mathcal{A}_1, \ldots, \mathcal{A}_k$ of algorithm $\mathcal{A}$ with the random strings $r_1, \ldots, r_k$, respectively.
(2) Let $\tilde{g} \leftarrow g(\perp)$ and denote $\varepsilon = \frac{1}{100}$ and $\varepsilon_0 = \frac{\varepsilon}{4\sqrt{2\lambda \ln(1/\delta)}}$.
(3) REPEAT at most λ times (outer loop)
  (a) Let $\hat{t} \leftarrow \frac{k}{2} + \mathrm{Lap}(\frac{2}{\varepsilon_0})$
  (b) REPEAT (inner loop)
    (i) Receive next update u_i.
    (ii) Insert update u_i into each algorithm $\mathcal{A}_1, \ldots, \mathcal{A}_k$ and obtain answers $y_{i,1}, \ldots, y_{i,k}$.
    (iii) If $|\{j : \tilde{g} \notin (1 \pm \frac{\alpha}{2}) \cdot y_{i,j}\}| + \mathrm{Lap}(\frac{4}{\varepsilon_0}) < \hat{t}$, then output estimate $\tilde{g}$ and CONTINUE inner loop. Otherwise, EXIT inner loop.
  (c) Recompute $\tilde{g} \leftarrow \mathtt{PrivateMed}(y_{i,1}, \ldots, y_{i,k})$, where `PrivateMed` is an $(\varepsilon_0, 0)$-differentially private algorithm for estimating the median of the data (see Theorem 2.6).
  (d) Output estimate $\tilde{g}$ and CONTINUE outer loop.

Proof. First observe that the algorithm samples at most $2m$ noises from the Laplace distribution with parameter ε_0 throughout its execution. By the properties of the Laplace distribution, with probability at least $1 - \delta$, it holds that *all* of these noises are at most $\frac{4}{\varepsilon_0} \log(\frac{2m}{\delta})$ in absolute value. We continue with the analysis assuming that this is the case.

For $i \in [m]$, let $\vec{u}_i = (u_1, \ldots, u_i)$ denote the stream consisting of the first i updates. Let $\mathcal{A}(r, \vec{u}_i)$ denote the estimate returned by the oblivious streaming algorithm $\mathcal{A}$ after the ith update, when it is executed with the random string r and receives the stream $\vec{u}_i$. Note that $y_{i,j} = \mathcal{A}(r_j, \vec{u}_i)$. Consider the following function:

$$f_{\vec{u}_i}(r) = \mathbb{1}\left\{\mathcal{A}(r, \vec{u}_i) \in \left(1 \pm \frac{\alpha}{10}\right) \cdot g(\vec{u}_i)\right\}.$$

By Lemma 3.1 the algorithm `RobustSketch` is $(\varepsilon = \frac{1}{100}, \delta)$-differentially private w.r.t. the collection of strings R. Furthermore, the updates in the stream $\vec{u}_i$ are chosen (by the adversary) by postprocessing the estimates returned by `RobustSketch`, and observe that the function $f_{\vec{u}_i}(\cdot)$ is defined by $\vec{u}_i$. As differential privacy is closed under postprocessing, we can view the *function* $f_{\vec{u}_i}(\cdot)$ as the outcome of a differentially private computation on the collection of strings R. Therefore, by the generalization properties of differential privacy (Theorem 2.8), assuming that $k \geq \frac{1}{\varepsilon^2} \log(\frac{2\varepsilon m}{\delta}) = \Theta(\log \frac{m}{\delta})$, with probability at least $(1 - \frac{\delta}{\varepsilon}) = 1 - O(\delta)$, for every $i \in [m]$ it holds that

$$\left| \mathbb{E}_r[f_{\vec{u}_i}(r)] - \frac{1}{k} \sum_{j=1}^{k} f_{\vec{u}_i}(r_j) \right| \leq 10\varepsilon = \frac{1}{10}.$$

We continue the analysis assuming that this is the case. Now observe that $\mathbb{E}_r[f_{\vec{u}_i}(r)] \geq 9/10$ by the utility guarantees of $\mathcal{A}$ (because when the stream is fixed, its answers are accurate to within multiplicative error of $(1 \pm \frac{\alpha}{10})$ with probability at least 9/10). Thus, for at least $(\frac{9}{10} - 10\varepsilon)k = 4k/5$ of the executions of $\mathcal{A}$, we have that $f_{\vec{u}_i}(r_j) = 1$, which means that $y_{i,j} \in (1 \pm \frac{\alpha}{10}) \cdot g(\vec{u}_i)$. In other words, in every timestep $i \in [m]$, we have that at least $4k/5$ of the $y_{i,j}$'s satisfy $y_{i,j} \in (1 \pm \frac{\alpha}{10}) \cdot g(\vec{u}_i)$.

We split the rest of the proof into the following two cases.

Case (a). If the algorithm outputs an estimate in step (3)(b)(iii), then, by our assumption on the magnitude of the noise, we have that

$$\left|\left\{j : \tilde{g} \in \left(1 \pm \frac{\alpha}{2}\right) \cdot y_{i,j}\right\}\right| \geq \frac{k}{2} - \frac{4}{\varepsilon_0} \log\left(\frac{2m}{\delta}\right) \geq \frac{4k}{10},$$

where the last inequality follows by asserting that

$$k = \Omega\left(\frac{1}{\varepsilon_0} \log \frac{m}{\delta}\right) = \Omega\left(\sqrt{\lambda \cdot \log\left(\frac{1}{\delta}\right)} \log\left(\frac{m}{\delta}\right)\right).$$

Thus, for at least $4k/5$ of the $y_{i,j}$'s, we have that $y_{i,j} \in (1 \pm \frac{\alpha}{10}) \cdot g(\vec{u}_i)$, and for at least $4k/10$ of them, we have that $\tilde{g} \in (1 \pm \frac{\alpha}{2}) \cdot y_{i,j}$. Therefore, there exists an index j that satisfies both conditions, in which case $\tilde{g} \in (1 \pm \alpha) \cdot g(\vec{u}_i)$, and the estimate we output is accurate.

Case (b). If the algorithm outputs an estimate in step (3)(d), then it is computed using algorithm `PrivateMed`, which is executed on the database $(y_{i,1}, \ldots, y_{i,k})$. By Theorem 2.6, assuming that[2]

$$k = \Omega\left(\frac{1}{\varepsilon_0} \log\left(\frac{\lambda}{\alpha\delta} \log m\right)\right) = \Omega\left(\sqrt{\lambda \cdot \log\left(\frac{1}{\delta}\right)} \cdot \log\left(\frac{\lambda}{\alpha\delta} \log m\right)\right),$$

then with probability at least $1 - \delta/\lambda$, algorithm `PrivateMed` returns an approximate median $\tilde{g}$ for the estimates $y_{i,1}, \ldots, y_{i,k}$, satisfying

$$\left|\left\{j : y_{i,j} \geq \tilde{g}\right\}\right| \geq \frac{4k}{10} \qquad \text{and} \qquad \left|\left\{j : y_{i,j} \leq \tilde{g}\right\}\right| \geq \frac{4k}{10}.$$

Since $4k/5$ of the $y_{i,j}$'s satisfy $y_{i,j} \in (1 \pm \frac{\alpha}{10}) \cdot g(\vec{u}_i)$, such an approximate median must also be in the range $(1 \pm \frac{\alpha}{10}) \cdot g(\vec{u}_i)$. This holds simultaneously for all of the estimates computed in step (3)(d) with probability at least $1 - \delta$. Note that in case (b), our estimate is actually accurate to within $(1 \pm \frac{\alpha}{10})$ rather than $(1 \pm \alpha)$.

Overall, with probability at least $1 - O(\delta)$, all of the estimates returned by the algorithm are accurate to within a multiplicative error of $(1 \pm \alpha)$. By increasing the constant hidden in the specification of k, we can boost the success probability to $1 - \delta$ as stated. □

We now show that, with high probability, the algorithm does not halt before the stream ends.

Lemma 3.3. *Let algorithm* `RobustSketch` *be executed with a parameter* $\lambda > \lambda_{\alpha/10,m}(g)$ *and* k *as defined in Lemma 3.2. Then with probability at least* $1 - \delta$*, the algorithm does not halt before the stream ends.*

Proof. As in the proof of Lemma 3.2, with probability at least $1 - \delta$, we have that

(1) All the Laplace noises sampled throughout the execution are at most $\frac{4}{\varepsilon_0} \log(\frac{2m}{\delta})$ in absolute value,
(2) All the estimates returned in step (3)(d) are accurate to within a multiplicative error of $(1 \pm \frac{\alpha}{10})$, and
(3) In every timestep $i \in [m]$, we have that at least $4k/5$ of the $y_{i,j}$'s satisfy $y_{i,j} \in (1 \pm \frac{\alpha}{10}) \cdot g(\vec{u}_i)$.

[2]We assume that the estimates that $\mathcal{A}$ returns are in the range $[-n^c, -1/n^c] \cup \{0\} \cup [1/n^c, n^c]$ (polynomially bounded in n) for some constant $c > 0$. In addition, before running `PrivateMed`, we may round each $y_{i,j}$ to its nearest power of $(1 + \frac{\alpha}{10})$, which has only a small effect on the error. There are at most $O(\frac{1}{\alpha} \log n)$ possible powers of $(1 + \frac{\alpha}{10})$ in that range, and hence `PrivateMed` guarantees error at most $\Gamma = O(\frac{1}{\varepsilon_0} \log(\frac{\lambda}{\alpha\delta} \log n))$ with probability at least $1 - \delta/\lambda$. See Theorem 2.6. Recall also our assumption that $\log m = \Theta(\log n)$.

We continue with the proof assuming that these statements hold. For $i \in [m]$, let $\tilde{g}_i$ denote the ith estimate that we output. Let $i_1 < i_2 \in [m]$ denote consecutive timesteps in which the algorithm outputs an estimate in step (3)(d) (and such that between i_1 and i_2, we compute the estimate in step (3)(b)(iii). Since we do not change our estimate between timesteps i_1 and i_2, we know that $\tilde{g}_{i_2-1} = \tilde{g}_{i_1}$.

Since in timestep i_2 we exit the inner loop (to output the estimate using step (3)(d)), a reasoning analogous to the one in case (a) of the proof of Lemma 3.2 (which uses the assertion on the value of k) implies that

$$\left|\left\{j : \tilde{g}_{i_2-1} \notin \left(1 \pm \frac{\alpha}{2}\right) \cdot y_{i_2,j}\right\}\right| \geq \frac{4k}{10}.$$

Since at least $4k/5$ of the $y_{i_2,j}$'s satisfy $y_{i_2,j} \in (1 \pm \frac{\alpha}{10}) \cdot g(\vec{u}_{i_2})$, there must exist a $y_{i_2,j}$ such that $\tilde{g}_{i_2-1} \notin (1 \pm \frac{\alpha}{2}) \cdot y_{i_2,j}$ and $y_{i_2,j} \in (1 \pm \frac{\alpha}{10}) \cdot g(\vec{u}_{i_2})$. Hence, $\tilde{g}_{i_2-1} \notin (1 \pm \frac{\alpha}{4}) \cdot g(\vec{u}_{i_2})$.

Recall (from case (b) of the proof of Lemma 3.2) that since at timestep i_1 we return the estimate $\tilde{g}_{i_1} = \tilde{g}_{i_2-1}$ using step (3)(d), then $\tilde{g}_{i_1} = \tilde{g}_{i_2-1} \in (1 \pm \frac{\alpha}{10}) \cdot g(\vec{u}_{i_1})$. Thus, we have established that $\tilde{g}_{i_2-1} \notin (1 \pm \frac{\alpha}{4}) \cdot g(\vec{u}_{i_2})$ and that $\tilde{g}_{i_2-1} \in (1 \pm \frac{\alpha}{10}) \cdot g(\vec{u}_{i_1})$, which means that

$$g(\vec{u}_{i_2}) \notin \left(1 \pm \frac{\alpha}{10}\right) \cdot g(\vec{u}_{i_1}).$$

It follows that every time we recompute $\tilde{g}$ in step (3)(d), it holds that the true value of g has changed by a multiplicative factor larger than $(1 + \frac{\alpha}{10})$ or smaller than $(1 - \frac{\alpha}{10})$. Therefore, the number of times we recompute $\tilde{g}$ in step (3)(d) cannot be bigger than $\lambda_{\alpha/10,m}(g)$. Thus, if the algorithm is executed with a parameter $\lambda > \lambda_{\alpha/10,m}(g)$, then with probability $1 - \delta$ the algorithm does not halt before the stream ends. □

The next theorem is obtained by combining Lemmas 3.2 and 3.3.

Theorem 3.4. *Let $\mathcal{A}$ be an oblivious streaming algorithm for a functionality g, which uses space $L(\frac{\alpha}{10}, \frac{1}{10})$ and guarantees accuracy $\frac{\alpha}{10}$ with success probability $\frac{9}{10}$ for streams of length m. Then there exists an adversarially robust streaming algorithm for g that guarantees accuracy α with success probability $1 - \delta$ for streams of length m using space*

$$O\left(L\left(\frac{\alpha}{10}, \frac{1}{10}\right) \cdot \sqrt{\lambda_{\frac{\alpha}{10},m}(g) \cdot \log\left(\frac{1}{\delta}\right)} \cdot \log\left(\frac{m}{\alpha\delta}\right)\right).$$

4 APPLICATIONS

Our algorithm can be applied to a wide range of streaming problems, such as estimating frequency moments, counting the number of distinct elements in the stream, identifying heavy-hitters in the stream, estimating the median of the stream, entropy estimation, and more. As an example, we now state the resulting bounds for F_2 estimation. Here, every update is a pair $u_i = (a_i, \Delta_i) \in [n] \times \mathbb{Z}$, and our goal is to keep track of the squared Euclidean norm of the frequencies vector, defined as follows.

Definition 4.1. The *frequency vector* of a stream $(a_1, \Delta_1), \ldots, (a_m, \Delta_m)$, where $(a_i, \Delta_i) \in ([n] \times \mathbb{Z})$, is the vector $f \in \mathbb{R}^n$ whose ℓth coordinate is

$$f_\ell = \sum_{i:a_i=\ell} \Delta_i.$$

We write $f^{(i)}$ to denote the frequency vector restricted to the first i updates.

In this section, we focus on estimating F_2, the second moment of the frequency vector. In other words, after every timestep i, we want to output an estimate of

$$\|f^{(i)}\|_2^2 = \sum_{\ell=1}^{n} \left|f_\ell^{(i)}\right|^2.$$

We use the following definition.

Definition 4.2 ([31]). Fix any $\tau \geq 1$. A data stream $(a_1, \Delta_1), \ldots, (a_m, \Delta_m)$, where $(a_i, \Delta_i) \in [n] \times \{1, -1\}$, is said to be an F_2 *τ-bounded deletion stream* if at every timestep $i \in [m]$ we have

$$\|f^{(i)}\|_2^2 \geq \frac{1}{\tau} \cdot \|h^{(i)}\|_2^2,$$

where h is the frequency vector of the stream with updates $(a_i, |\Delta_i|)$.

The following lemma relates the bounded deletion parameter τ to the flip number of the stream.

Lemma 4.3 ([10]). *The $\lambda_{\alpha,m}(\|\cdot\|_2^2)$ flip number of a τ-bounded deletion stream is at most $O(\frac{\tau}{\alpha^2}\log m)$.*

The following theorem is now obtained by applying algorithm `RobustSketch` with the oblivious algorithm of Kane et al. [32] that uses space $O(\frac{1}{\alpha^2}\log^2(\frac{m}{\delta}))$.

Theorem 4.4. *There is an adversarially robust F_2 estimation algorithm for τ-bounded deletion streams of length m that guarantees α accuracy with probability at least $1 - \frac{1}{m}$. The space used by the algorithm is*

$$O\left(\frac{\sqrt{\tau}}{\alpha^3} \cdot \log^4(m)\right).$$

In contrast, the F_2 estimation algorithm of [10] for τ-bounded deletion streams uses space $O(\frac{\tau}{\alpha^4} \cdot \log^3(m))$. Specifically, the space bound of Ben-Eliezer et al. [10] grows as $\frac{\tau}{\alpha^4}$, whereas ours only grows as $\frac{\sqrt{\tau}}{\alpha^3}$ (at the cost of additional $\log(m)$ factors). As we mentioned, our results are also meaningful for the insertion-only model. Specifically, we have the following lemma.

Lemma 4.5 ([10]). *The $\lambda_{\alpha,m}(\|\cdot\|_2^2)$ flip number of an insertion-only stream is at most $O(\frac{1}{\alpha}\log m)$.*

The following theorem is obtained by applying algorithm `RobustSketch` with the oblivious algorithm of Blasiok et al. [12] that uses space $\tilde{O}(\frac{1}{\alpha^2}\log(m)\log(\frac{1}{\delta}))$.

Theorem 4.6. *There is an adversarially robust F_2 estimation algorithm for insertion-only streams of length m that guarantees α accuracy with probability at least $1-\frac{1}{m}$. The space used by the algorithm is*

$$\tilde{O}\left(\frac{1}{\alpha^{2.5}} \cdot \log^4(m)\right).$$

In contrast, the F_2 estimation algorithm of Ben-Eliezer et al. [10] for insertion-only streams uses space $\tilde{O}(\frac{1}{\alpha^3} \cdot \log^2(m))$. Our bound, therefore, improves the space dependency on α (at the cost of additional logarithmic factors).

APPENDIX

A AN EXTENDED MODEL

Our results are applicable to a setting where the `StreamingAlgorithm` is not required to release an estimate in every timestep but only at T timesteps (chosen adaptively by the adversary and referred to as "query times"). Formally, we consider the following variant of the game specified in Section 2.1, in which m denotes the length of the stream and T denotes the number of allowed queries:

(1) For round $i = 1, 2, \ldots, m$
   (a) The `Adversary` chooses an update u_i and a query demand $q_i \in \{0, 1\}$, under the restriction that $\sum_{j=1}^{i} q_j \leq T$.
   (b) The `StreamingAlgorithm` processes the new update u_i. If $q_i = 1$, then the `StreamingAlgorithm` outputs a response z_i.

In other words, throughout the execution the `Adversary` determines the updates in the stream adaptively, and it is allowed to query the `StreamingAlgorithm` in at most T timesteps of its choice. The goal of the `Adversary` is to make the `StreamingAlgorithm` output an incorrect response z_i at some query time i in the stream. Our results extend to this setting, yielding the following result.

Theorem A.1. *Fix any function g and fix $\alpha, \delta > 0$. Let $\mathcal{A}$ be an oblivious streaming algorithm for g that uses space $L(\frac{\alpha}{10}, \frac{1}{10})$ and guarantees accuracy $\frac{\alpha}{10}$ with success probability $\frac{9}{10}$ for streams of length m. Then there exists an adversarially robust streaming algorithm for g that guarantees accuracy α with success probability $1 - \delta$ for streams of length m using space*

$$O\left(L\left(\frac{\alpha}{10}, \frac{1}{10}\right) \cdot \sqrt{\min\left\{T\ ,\ \lambda_{\frac{\alpha}{10}, m}(g)\right\} \cdot \log\left(\frac{1}{\delta}\right)} \cdot \log\left(\frac{m}{\alpha\delta}\right)\right).$$

Our techniques also extend to cases where the underlying target function g is *not* a real valued function, e.g., where the goal is to identify the heavy-hitters in the stream, provided that there is an efficient way to privately aggregate intermediate outputs for the problem at hand.

ACKNOWLEDGMENTS

The authors are grateful to Amos Beimel and Edith Cohen for many helpful discussions.

REFERENCES

[1] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. 2012. Analyzing graph structure via linear measurements. In *Proceedings of SODA*. 459–467. https://doi.org/10.1137/1.9781611973099.40
[2] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. 2012. Graph sketches: Sparsification, spanners, and subgraphs. In *Proceedings of PODS*. 5–14. https://doi.org/10.1145/2213556.2213560
[3] Noga Alon, Yossi Matias, and Mario Szegedy. 1999. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.* 58, 1 (1999), 137–147. https://doi.org/10.1006/jcss.1997.1545
[4] Idan Attias, Edith Cohen, Moshe Shechner, and Uri Stemmer. 2021. A framework for adversarial streaming via differential privacy and difference estimators. *CoRR* abs/2107.14527. https://arxiv.org/abs/2107.14527.
[5] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. 2002. Counting distinct elements in a data stream. In *Proceedings of RANDOM*. 1–10.
[6] Raef Bassily, Kobbi Nissim, Adam D. Smith, Thomas Steinke, Uri Stemmer, and Jonathan Ullman. 2016. Algorithmic stability for adaptive data analysis. In *Proceedings of STOC*. 1046–1059.
[7] Amos Beimel, Iftach Haitner, Nikolaos Makriyannis, and Eran Omri. 2018. Tighter bounds on multi-party coin flipping via augmented weak martingales and differentially private sampling. In *Proceedings of FOCS*. 838–849.
[8] Amos Beimel, Kobbi Nissim, and Uri Stemmer. 2013. Private learning and sanitization: Pure vs. approximate differential privacy. In *Proceedings of APPROX-RANDOM*. 363–378.
[9] Omri Ben-Eliezer, Talya Eden, and Krzysztof Onak. 2022. Adversarially robust streaming via dense-sparse trade-offs. In *Proceedings of SOSA@SODA*. 214–227.
[10] Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. 2022. A framework for adversarially robust streaming algorithms. *J. ACM* 69, 2 (2022), Article 17, 33 pages.
[11] Omri Ben-Eliezer and Eylon Yogev. 2020. The adversarial robustness of sampling. In *Proceedings of PODS*. 49–62.
[12] Jaroslaw Blasiok, Jian Ding, and Jelani Nelson. 2017. Continuous monitoring of l_p norms in data streams. In *Proceedings of APPROX/RANDOM*. Article 32, 13 pages.
[13] Mark Bun, Cynthia Dwork, Guy N. Rothblum, and Thomas Steinke. 2018. Composable and versatile privacy via truncated CDP. In *Proceedings of STOC*. 74–86.
[14] Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. 2015. Differentially private release and learning of threshold functions. In *Proceedings of FOCS*. 634–649.

[15] Moses Charikar, Kevin Chen, and Martin Farach-Colton. 2002. Finding frequent items in data streams. In *Proceedings of ICALP*. 693–703.
[16] Graham Cormode and S. Muthukrishnan. 2005. An improved data stream summary: The count-min sketch and its applications. *J. Algorithms* 55, 1 (2005), 58–75. https://doi.org/10.1016/j.jalgor.2003.12.001
[17] Graham Cormode and S. Muthukrishnan. 2005. What's hot and what's not: Tracking most frequent items dynamically. *ACM Trans. Database Syst.* 30, 1 (2005), 249–278. https://doi.org/10.1145/1061318.1061325
[18] Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 2002. Maintaining stream statistics over sliding windows. *SIAM J. Comput.* 31, 6 (2002), 1794–1813. https://doi.org/10.1137/S0097539701398363
[19] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth. 2015. Preserving statistical validity in adaptive data analysis. In *Proceedings of STOC*. 117–126.
[20] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Proceedings of TCC*. 265–284.
[21] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. 2009. On the complexity of differentially private data release: Efficient algorithms and hardness results. In *Proceedings of STOC*. 381–390.
[22] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. 2010. Boosting and differential privacy. In *Proceedings of FOCS*. 51–60.
[23] Michael Elkin and Jian Zhang. 2006. Efficient algorithms for constructing $(1+\epsilon, \beta)$-spanners in the distributed and streaming models. *Distrib. Comput.* 18, 5 (2006), 375–385. https://doi.org/10.1007/s00446-005-0147-2
[24] Philippe Flajolet and G. Nigel Martin. 1985. Probabilistic counting algorithms for database applications. *J. Comput. System Sci.* 31, 2 (1985), 182–209. https://doi.org/10.1016/0022-0000(85)90041-8
[25] A. C. Gilbert, B. Hemenway, A. Rudra, M. J. Strauss, and M. Wootters. 2012. Recovering simple signals. In *Proceedings of ITA*. 382–391.
[26] A. C. Gilbert, B. Hemenway, M. J. Strauss, D. P. Woodruff, and M. Wootters. 2012. Reusable low-error compressive sampling schemes through privacy. In *Proceedings of SSP*. 536–539.
[27] Moritz Hardt and Guy N. Rothblum. 2010. A multiplicative weights mechanism for privacy-preserving data analysis. In *Proceedings of FOCS*. 61–70.
[28] Moritz Hardt and Jonathan Ullman. 2014. Preventing false discovery in interactive data analysis is hard. In *Proceedings of FOCS*. 454–463.
[29] Moritz Hardt and David P. Woodruff. 2013. How robust are linear sketches to adaptive inputs? In *Proceedings of STOC*. 121–130.
[30] Piotr Indyk and David Woodruff. 2005. Optimal approximations of the frequency moments of data streams. In *Proceedings of STOC*. 202–208.
[31] Rajesh Jayaram and David P. Woodruff. 2018. Data streams with bounded deletions. In *Proceedings of PODS*. 341–354.
[32] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. 2010. On the exact space complexity of sketching and streaming small norms. In *Proceedings of SODA*. 1161–1178.
[33] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. 2010. An optimal algorithm for the distinct elements problem. In *Proceedings of PODS*. 41–52.
[34] Haim Kaplan, Katrina Ligett, Yishay Mansour, Moni Naor, and Uri Stemmer. 2020. Privately learning thresholds: Closing the exponential gap. In *Proceedings of COLT*. 2263–2285.
[35] Haim Kaplan, Yishay Mansour, Kobbi Nissim, and Uri Stemmer. 2021. Separating adaptive streaming from oblivious streaming using the bounded storage model. In *Proceedings of CRYPTO*. 94–121.
[36] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. 2017. Accessing data while preserving privacy. *CoRR* abs/1706.01552. http://arxiv.org/abs/1706.01552.
[37] Gurmeet Singh Manku and Rajeev Motwani. 2002. Approximate frequency counts over data streams. In *Proceedings of VLDB*. 346–357.
[38] Frank McSherry and Kunal Talwar. 2007. Mechanism design via differential privacy. In *Proceedings of FOCS*. 94–103.
[39] Ilya Mironov, Moni Naor, and Gil Segev. 2011. Sketching in adversarial environments. *SIAM J. Comput.* 40, 6 (2011), 1845–1870. https://doi.org/10.1137/080733772
[40] S. Muthukrishnan. 2005. Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci.* 1, 2 (2005), 117–236. https://doi.org/10.1561/0400000002
[41] Jelani Nelson. 2011. *Sketching and Streaming Algorithms*. Ph. D. Dissertation. Massachusetts Institute of Technology, Cambridge, MA. http://hdl.handle.net/1721.1/66314.
[42] Kobbi Nissim, Adam D. Smith, Thomas Steinke, Uri Stemmer, and Jonathan Ullman. 2018. The limits of post-selection generalization. In *Proceedings of NeurIPS*. 6402–6411.
[43] Kobbi Nissim and Uri Stemmer. 2019. Concentration bounds for high sensitivity functions through differential privacy. *J. Priv. Confid.* 9, 1 (2019), 1–33. https://doi.org/10.29012/jpc.658

[44] Thomas Steinke and Jonathan Ullman. 2015. Interactive fingerprinting codes and the hardness of preventing false discovery. In *Proceedings of COLT*. 1588–1628.
[45] Uri Stemmer. 2016. *Individuals and Privacy in the Eye of Data Analysis*. Ph. D. Dissertation. Ben-Gurion University of the Negev.
[46] David P. Woodruff and Samson Zhou. 2022. Tight bounds for adversarially robust streams and sliding windows via difference estimators. In *Proceedings of FOCS*. 1183–1196.

Received 29 June 2021; revised 31 July 2022; accepted 31 July 2022