A Framework for Adversarially Robust Streaming Algorithms

Omri Ben-Eliezer* Rajesh Jayaram† David P. Woodruff† Eylon Yogev‡

Abstract

We investigate the adversarial robustness of streaming algorithms. In this context, an algorithm is considered robust if its performance guarantees hold even if the stream is chosen adaptively by an adversary that observes the outputs of the algorithm along the stream and can react in an online manner. While deterministic streaming algorithms are inherently robust, many central problems in the streaming literature do not admit sublinear-space deterministic algorithms; on the other hand, classical space-efficient randomized algorithms for these problems are generally not adversarially robust. This raises the natural question of whether there exist efficient adversarially robust (randomized) streaming algorithms for these problems.

In this work, we show that the answer is positive for various important streaming problems in the insertion-only model, including distinct elements and more generally F_p-estimation, F_p-heavy hitters, entropy estimation, and others. For all of these problems, we develop adversarially robust $(1+\varepsilon)$-approximation algorithms whose required space matches that of the best known non-robust algorithms up to a $\mathrm{poly}(\log n, 1/\varepsilon)$ multiplicative factor (and in some cases even up to a constant factor). Towards this end, we develop several generic tools allowing one to efficiently transform a non-robust streaming algorithm into a robust one in various scenarios.

1 Introduction

The streaming model of computation is a central and crucial tool for the analysis of massive datasets, where the sheer size of the input imposes stringent restrictions on the memory, computation time, and other resources available to the algorithms. Examples of theoretical and practical settings where streaming algorithms are in need are easy to encounter. These include internet routers and traffic logs, databases, sensor networks, financial transaction data, and scientific data streams. Given this wide range of applicability, there has been significant effort devoted to designing and analyzing extremely efficient one-pass algorithms. We recommend the survey of [33] for a comprehensive overview of streaming algorithms and their applications.

Many central problems in the streaming literature do not admit sublinear-space deterministic algorithms, and in these cases randomized solutions are necessary. In other cases, randomized solutions are more efficient and simpler to implement than their deterministic counterparts. While randomized streaming algorithms are well-studied, the vast majority of them are defined and analyzed in the *static* setting, where the stream is worst-case but fixed in advance, and only then the randomness of the algorithm is chosen. However, assuming that the stream sequence is independent of the chosen randomness, and in particular that future elements of the stream do not depend on previous outputs of the streaming algorithm, may not be realistic [31, 15, 16, 20, 34, 5], even in non-adversarial settings. For example, suppose that a user sequentially makes queries to a database, and receives an immediate response after each query. Naturally, future queries made by the user in such a setting may heavily depend on the responses

*Tel Aviv University.

†Carnegie Mellon University. Supported by the National Science Foundation under Grant No. CCF-1815840.

‡Boston University and Tel Aviv University. Supported by the ISF grants 484/18, 1789/19, Len Blavatnik and the Blavatnik Foundation, and The Blavatnik Interdisciplinary Cyber Research Center at Tel Aviv University.

given by the database to previous queries. In other words, the stream updates are chosen adaptively, and cannot be assumed to be fixed in advance.

A streaming algorithm that works even when the stream is adaptively chosen by an adversary (the precise definition given next) is said to be *adversarially robust*. Deterministic algorithms are inherently adversarially robust, since they are guaranteed to be correct on all possible inputs. However, the large gap in performance between deterministic and randomized streaming algorithms for many problems motivates the need for designing adversarially robust randomized algorithms, if they even exist. In particular, we would like to design adversarially robust randomized algorithms which are as space and time efficient as their static counterparts, and yet as robust as deterministic algorithms. The study of such algorithms is the main focus of our work.

The Adversarial Setting. There are several ways to define the adversarial setting, which depends on the information the adversary (who chooses the stream) can observe from the streaming algorithm, as well as other restrictions imposed on the adversary. For the most part, we consider a general model, where the adversary is allowed unbounded computational power and resources, though we do discuss the case later when the adversary is computationally bounded. At each point in time, the streaming algorithm publishes its output to a query for the stream. The adversary observes these outputs one-by-one, and can choose the next update to the stream adaptively, depending on the full history of the outputs and stream updates. The goal of the adversary is to force the streaming algorithm to eventually produce an *incorrect* output to the query, as defined by the specific streaming problem in question.[1]

Formally, a data stream of length m over a domain $[n]$ is a sequence of updates of the form $(a_1, \Delta_1), (a_2, \Delta_2), \ldots, (a_m, \Delta_m)$ where $a_t \in [n]$ is an index and $\Delta_t \in \mathbb{Z}$ is an increment or decrement to that index. The *frequency vector* $f \in \mathbb{R}^n$ of the stream is the vector with i^{th} coordinate $f_i = \sum_{t:a_t=i} \Delta_t$. We write $f^{(t)}$ to denote the frequency vector restricted to the first t updates, namely $f_i^{(t)} = \sum_{j \le t:a_j=i} \Delta_j$. It is assumed at all points t that $\|f^{(t)}\|_\infty \le M$ for some $M > 0$, and that $\log(mM) = O(\log n)$. In the *insertion-only* model, the updates are assumed to be positive, meaning $\Delta_t > 0$, whereas in the *turnstile* model Δ_t can be positive or negative.

The general task in streaming is to respond to some query $\mathcal{Q}$ about the frequency vector $f^{(t)}$ at each point in time $t \in [m]$. Oftentimes, this query is to approximate[2] some function $g : \mathbb{R}^n \to \mathbb{R}$ of $f^{(t)}$. For example, counting the number of distinct elements in a data stream is among the most fundamental problems in the streaming literature; here $g(f^{(t)})$ is the number of non-zero entries in $f^{(t)}$. Since exact computation cannot be done in sublinear space [9], the goal is to approximate the value of $g(f^{(t)})$ to within a multiplicative factor of $(1 \pm \varepsilon)$. Another important streaming problem (which is not directly an estimation task) is the *Heavy-Hitters* problem, where the algorithm is tasked with finding all the coordinates in $f^{(t)}$ which are larger than some threshold τ.

Formally, the adversarial setting is modeled by a two-player game between a (randomized) StreamingAlgorithm and an Adversary. At the beginning, a query $\mathcal{Q}$ is fixed, which the StreamingAlgorithm must continually reply to. The game proceeds in rounds, where in the t-th round:

1. The Adversary chooses an update $u_t = (a_t, \Delta_t)$ for the stream, which can depend, in particular, on all previous stream updates and outputs of StreamingAlgorithm.

[1] In the streaming literature, an algorithm is often required to be correct on a query made only *once*, at the end of the stream. This is a *one-shot* guarantee, as opposed to the *tracking* guarantee as defined here. However, the two settings are nearly equivalent. Indeed, for almost all streaming problems, a one-shot algorithm can be made into a tracking algorithm with at most an $O(\log n)$ blow-up in space, by simply setting the failure probability small enough to union bound over all points in the stream.

[2] Ideally, one might wish to exactly compute the function g; however, in many cases, and in particular for the problems that we consider here, exact computation cannot be done with sublinear space.

2. The StreamingAlgorithm processes the new update u_t and outputs its current response R^t to the query $\mathcal{Q}$.

3. The Adversary observes R^t (stores it) and proceeds to the next round.

The goal of the Adversary is to make the StreamingAlgorithm output an incorrect response R^t to $\mathcal{Q}$ at some point t in the stream. For example, in the distinct elements problem, the adversary's goal is that on some step t, the estimate R^t will fail to be a $(1+\varepsilon)$-approximation of the true current number of distinct elements $|\{i \in [n] : f_i^{(t)} \neq 0\}|$.

Streaming algorithms in the adversarial setting. It was shown by Hardt and Woodruff [20] that linear sketches are inherently *non-robust* in adversarial settings for a large family of problems, thus demonstrating a major limitation of such sketches. In particular, their results imply that no linear sketch can approximate the Euclidean norm of its input to within a polynomial multiplicative factor in the adversarial (turnstile) setting. Here, a linear sketch is an algorithm whose output depends only on values Af and A, for some (usually randomized) sketching matrix $A \in \mathbb{R}^{k \times n}$. This is quite unfortunate, as the vast majority of turnstile streaming algorithms are in fact linear sketches.

On the positive side, a recent work of Ben-Eliezer and Yogev [5] showed that *random sampling* is quite robust in the adaptive adversarial setting, albeit with a slightly larger sample size. While uniform sampling is a rather generic and important tool, it is not sufficient for solving many important streaming tasks, such as estimating frequency moments (F_p-estimation), finding L_2 heavy hitters, and various other central data analysis problems. This raises the natural question of whether there exist efficient adversarially robust randomized streaming algorithms for these problems and others, which is the main focus of this work. Perhaps even more importantly, we ask the following.

Is there a generic technique to transform a static streaming algorithm into an adversarially robust streaming algorithm?

This work answers the above questions affirmatively for a large class of algorithms.

1.1 Our Results

We devise adversarially robust algorithms for various fundamental insertion-only streaming problems, including distinct element estimation, F_p moment estimation, heavy hitters, entropy estimation, and several others. In addition, we give adversarially robust streaming algorithms which can handle a bounded number of deletions as well. The required space of our adversarially robust algorithms matches that of the best known non-robust ones up to a small multiplicative factor. In contrast, we demonstrate that some classical randomized algorithms for streaming problems in the static setting, such as the celebrated Alon-Matias-Szegedy (AMS) sketch [3] for F_2-estimation, are inherently non-robust to adaptive adversarial attacks in a strong sense.

Our adversarially robust algorithms make use of two generic robustification frameworks that we develop, allowing one to efficiently transform a non-robust streaming algorithm into a robust one in various settings. Both of the robustification methods rely on the fact that functions of interest do not drastically change their value too many times along the stream. We combine this observation with a rounding technique so as to avoid leaking information to the adaptive adversary.

The first method, called *sketch switching*, maintains multiple instances of the non-robust algorithm and switches between them in a way that cannot be exploited by the adversary. The second technique bounds the number of *computation paths* possible in the two-player adversarial game. This technique maintains only one copy of a non-robust algorithm, albeit with an extremely small probability of error δ. We show that a carefully rounded sequence of outputs

Problem	Static Rand.	Deterministic	**Adversarial**	Comments
Distinct elements (F_0)	$\tilde{O}(\varepsilon^{-2}+\log n)$ [6]	$\Omega(n)$ [9]	$\boldsymbol{\tilde{O}(\varepsilon^{-3}+\log n)}$	
			$\boldsymbol{\tilde{O}(\varepsilon^{-2}+\log n)}$	crypto/random oracle
F_p est., $p \in (0,2] \setminus \{1\}$	$O(\varepsilon^{-2}\log n)$ [7]	$\tilde{\Omega}(2^{-1/(1-p)}n)$ [9]	$\boldsymbol{\tilde{O}(\varepsilon^{-3}\log n)}$	
	$O(\varepsilon^{-3}\log^2 n)$ [27]		$\boldsymbol{\tilde{O}(\varepsilon^{-3}\log^3 n)}$	$\delta = \Theta(n^{-(1/\varepsilon)\log n})$
F_p estimation, $p > 2$	$O\big(n^{1-\frac{2}{p}}(\varepsilon^{-3}\log^2 n + \varepsilon^{-\frac{6}{p}}\log^{\frac{4}{p}+1} n)\big)$ [14]	$\Omega(n)$ [9]	$\boldsymbol{O\big(n^{1-\frac{2}{p}}(\varepsilon^{-3}\log^2 n + \varepsilon^{-\frac{6}{p}}\log^{\frac{4}{p}+1} n)\big)}$	$\delta = \Theta(n^{-(1/\varepsilon)\log n})$
ℓ_2 Heavy Hitters	$O(\varepsilon^{-2}\log^2 n)$ [8]	$\Omega(\sqrt{n})$ [26]	$\boldsymbol{\tilde{O}(\varepsilon^{-3}\log^2 n)}$	
Entropy Estimation	$O(\varepsilon^{-2}\log^3 n)$ [11]	$\tilde{\Omega}(n)$	$\boldsymbol{O(\varepsilon^{-5}\log^6 n)}$	
	$\tilde{O}(\varepsilon^{-2})$ [23]		$\boldsymbol{O(\varepsilon^{-5}\log^4 n)}$	crypto/random oracle
Turnstile F_p, $p \in (0,2]$	$O(\varepsilon^{-2}\lambda\log^2 n)$ [27]	$\Omega(n)$ [3]	$\boldsymbol{O(\varepsilon^{-2}\lambda\log^2 n)}$	λ-bounded F_p flip number, $\delta = \Theta(n^{-\lambda})$
F_p, $p \in [1,2]$, α-bdd. del.	$\tilde{O}(\varepsilon^{-2}\log\alpha\log n + \log^2 n)$ [22]	$\tilde{\Omega}(2^{-1/(1-p)}n)$ [9]	$\boldsymbol{O(\alpha\varepsilon^{-(2+p)}\log^3 n)}$	static only for $p = 1$

Table 1: A summary of our adversarially robust algorithms (in bold), as compared to the best known upper bounds for randomized algorithms in the static setting and lower bounds for deterministic algorithms. Note that all stated algorithms provide tracking. All results except for the last two (which hold in restricted versions of the turnstile model) are for insertion only streams. We write $\tilde{O}, \tilde{\Omega}$ to hide $\log \varepsilon^{-1}$ and $\log\log n$ factors. The lower bound for deterministic entropy estimation follows from a reduction from estimating F_p for $p = 1 + \tilde{\Theta}(\frac{\varepsilon}{\log^2 n})$ to entropy estimation [21].

generates only a small number of possible computation paths, which can then be used to ensure robustness by union bounding over these paths. The framework is described in Section 3.

The two above methods are incomparable: for some streaming problems the former is more efficient, while for others, the latter performs better, and we show examples of each. Specifically, sketch switching can exploit efficiency gains of *strong-tracking*, resulting in particularly good performance for static algorithms that can respond correctly to queries at each step without having to union bound over all m steps. In contrast, the computation paths technique can exploit an algorithm with good dependency on δ (the failure probability). Namely, algorithms that have small dependency in update-time or space on δ will benefit from the computation paths technique.

For each of the problems we consider, we show how to use the framework, in addition to some further techniques which we develop along the way, to solve it. Interestingly, we also demonstrate how cryptographic assumptions (which were not commonly used before in the streaming context) can be applied to obtain an adversarially robust algorithm against computationally bounded adversaries for the distinct elements problem at essentially no extra cost over the space optimal non-robust one. See Table 1 for a summary of our results in the adversarial setting compared to the state-of-the-art in the static setting, as well as to deterministic algorithms.

Distinct elements and F_p-estimation. Our first suite of results provides robust streaming algorithms for estimating F_p, the p^{th} frequency moment of the frequency vector, defined as $F_p = \|f\|_p^p = \sum_{i=1}^n |f_i|^p$, where we interpret $0^0 = 0$. Estimating frequency moments has a myriad of applications in databases, computer networks, data mining, and other contexts. Efficient algorithms for estimating distinct elements (i.e., estimating F_0) are important for databases, since query optimizers can use them to find the number of unique values of an attribute without having to perform an expensive sort on the values. Efficient algorithms for F_2 are useful for determining the output size of self-joins in databases, and for computing the surprise index of a data sequence [18]. Higher frequency moments are used to determine data skewness, which is important in parallel database applications [12].

We remark that for any fixed $p \neq 1$,[3] including $p = 0$, any deterministic insertion-only algorithm for F_p-estimation requires $\Omega(n)$ space [9]. In contrast, we will show that randomized adversarially robust algorithms exist for all p, whose space complexity either matches or has a

[3]Note that there is a trivial $O(\log n)$-bit insertion only F_1 estimation algorithm: keeping a counter for $\sum_t \Delta_t$.

small multiplicative overhead over the best static randomized algorithms.

We begin with several results on the problem of estimating distinct elements, or F_0 estimation. The first of them utilizes an optimized version of the sketch switching method to derive an upper bound. The result is an adversarially robust F_0 estimation algorithm whose complexity is only a $\Theta(\frac{1}{\varepsilon}\log\varepsilon^{-1})$ factor larger than the optimal static (non-robust) algorithm [6].

Theorem 1.1 (Robust Distinct Elements by Sketch Switching; see Theorem 5.1)**.** *There is an algorithm which, when run on an adversarial insertion only stream, with probability at least $1-\delta$ produces at every step $t \in [m]$ an estimate R^t such that $R^t = (1\pm\varepsilon)\|f^{(t)}\|_0$. The space used by the algorithm is*

$$O\left(\frac{\log(1/\varepsilon)}{\varepsilon}\left(\frac{\log\varepsilon^{-1}+\log\delta^{-1}+\log\log n}{\varepsilon^2}+\log n\right)\right).$$

The second result utilizes a different approach, by applying the computation paths method. The space complexity is slightly worse, which is a result of setting the failure probability $\delta < n^{-\frac{1}{\varepsilon}\log n}$ for any given static algorithm. However, we introduce a new *static* algorithm for F_0 estimation which has very small update-time dependency on δ, and nearly optimal space complexity. As a result, by applying our computation paths method to this new static algorithm, we obtain an adversarially robust F_0 estimation algorithm with extremely fast update time (note that the update time of the above sketch switching algorithm would be $O(\varepsilon^{-1}\log n)$ to obtain the same result, even for constant δ).

Theorem 1.2 (Fast Adversarially Robust Distinct Elements, see Theorem 5.4)**.** *There is a streaming algorithm which, with probability $1-n^{-(C/\varepsilon)\log n}$ for any constant $C \geq 1$, when run on an adversarial chosen insertion-only data stream, returns a $(1\pm\varepsilon)$ multiplicative estimate of the number of distinct elements at every step in the stream. The space required is $O(\frac{1}{\varepsilon^3}\log^3 n)$, and the algorithm runs in $O((\log^2\log\frac{\log n}{\varepsilon})\cdot(\log\log\log\frac{\log n}{\varepsilon}))$ worst case time per update.*

The third result takes a different approach: it shows that under certain standard cryptographic assumptions, there exists an adversarially robust algorithm which asymptotically matches the space complexity of the best non-robust tracking algorithm for distinct elements. The cryptographic assumption is that an exponentially secure pseudorandom function exists (in practice one can take, for instance, AES as such a function). While our other algorithms in this paper hold even against an adversary which is unbounded computationally, in this particular result we assume that the adversary runs in polynomial time. See Section 10 for more details.

Theorem 1.3 (Robust Distinct Elements by Cryptographic Assumptions; see Theorem 10.1)**.** *In the random oracle model, there is an F_0-estimation (tracking) streaming algorithm in the adversarial setting, that for an approximation parameter ε uses $O(\varepsilon^{-2}(\log 1/\varepsilon+\log\log n)+\log n)$ bits of memory, and succeeds with probability $3/4$.*

Moreover, under a suitable cryptographic assumption, assuming the adversary has bounded running time of n^c, where m is the stream length and c is a constant, the random oracle can be replaced with a concrete function and the total memory is $O(\varepsilon^{-2}(\log 1/\varepsilon+\log\log n)+c\log n)$.

Here, the random oracle model means that the algorithm is given read access to an arbitrarily long string of random bits.

Our next set of results provides adversarially robust algorithms for F_p-estimation with $p > 0$. The following result concerns the case $0 < p \leq 2$. It was previously shown [9] that for p bounded away from one, $\Omega(n)$ space is required to deterministically estimate $\|f\|_p^p$, even in the insertion only model [9]. On the other hand, space-efficient non-robust randomized algorithms for F_p-estimation exist. We leverage these, along with an optimized version of the sketch switching technique to save a $\log n$ factor, and obtain the following.

Theorem 1.4 (F_p-estimation for $0 < p \leq 2$; see Theorem 4.1)**.** *Fix $0 < \varepsilon, \delta \leq 1$ and $0 < p \leq 2$. There is a streaming algorithm in the insertion-only adversarial model which outputs at each step a value R^t such that $R^t = (1 \pm \varepsilon)\|f^{(t)}\|_p$ at every step $t \in [m]$, and succeeds with probability $1 - \delta$. The algorithm uses $O(\varepsilon^{-3} \log n \log \varepsilon^{-1}(\log \varepsilon^{-1} + \log \delta^{-1} + \log \log n))$ bits of space.*

We remark that the space complexity of Theorem 1.4 is within a $\Theta(\varepsilon^{-1} \log \varepsilon^{-1})$ factor of the best known static (non-robust) algorithm [7] . While for most values of δ, the above theorem using sketch switching has better space complexity than the computation paths reduction, for the regime of very small failure probability δ it is actually preferable to use the latter, as we now state.

Theorem 1.5 (F_p estimation for small δ; see Theorem 4.2)**.** *Fix any $0 < \varepsilon < 1$, $0 < p \leq 2$, and $\delta < n^{-C\frac{1}{\varepsilon}\log n}$ for a sufficiently large constant $C > 1$. There is a streaming algorithm for the insertion-only adversarial model which, with probability $1 - \delta$, successfully outputs at each step $t \in [m]$ a value R^t such that $R^t = (1 \pm \varepsilon)\|f^{(t)}\|_p$. The space used by the algorithm is $O\left(\frac{1}{\varepsilon^2} \log n \log \delta^{-1}\right)$ bits.*

In addition, we show that for turnstile streams with bounded F_p *flip number* (defined formally in Section 3), efficient adversarially robust algorithms exist. Roughly speaking, the F_p flip number is the number of times that the F_p moment changes by a factor of $(1 + \varepsilon)$. Our algorithms have extremely small failure probability of $\delta = n^{-\lambda}$, and have optimal space among turnstile algorithms with this value of δ [25].

Theorem 1.6 (F_p Estimation for λ-flip number turnstile streams; See Theorem 4.3)**.** *Let $\mathcal{S}_\lambda$ be the set of all turnstile streams with F_p flip number at most $\lambda \geq \lambda_{\varepsilon,m}(\|\cdot\|_p^p)$ for any $0 < p \leq 2$. Then there is an adversarially robust streaming algorithm for the class $\mathcal{S}_\lambda$ of streams that, with probability $1 - n^{-C\lambda}$ for any constant $C > 0$, outputs at each time step a value R^t such that $R^t = (1 \pm \varepsilon)\|f\|_p^p$. The space used by the algorithm is $O(\varepsilon^{-2}\lambda \log^2 n)$.*

The next result concerns F_p-estimation for $p > 2$. Here again, we provide an adversarially robust algorithm which is optimal up to a small multiplicative factor. This result applies the computation paths robustification method as a black box. Notably, a classic lower bound of [4] shows that for $p > 2$, $\Omega(n^{1-2/p})$ space is required to estimate $\|f\|_p^p$ up to a constant factor (improved lower bounds have been provided since, e.g., [29]). By using our computation paths technique, we obtain adversarially robust F_p moment estimation algorithms as well for $p > 2$.

Theorem 1.7 (Robust F_p estimation for $p > 2$; see Theorem 4.4)**.** *Fix any $\varepsilon > 0$, and fix any $p > 2$. There is a streaming algorithm for the insertion-only adversarial model which, with probability $1 - n^{-(c \log n)/\varepsilon}$ for any constant $c > 1$, successfully outputs at each step a value R^t such that $R^t = (1 \pm \varepsilon)\|f^{(t)}\|_p$ at every step $t \in [m]$. The space used by the algorithm is*

$$O\left(n^{1-2/p}\left(\varepsilon^{-3}\log^2 n + \varepsilon^{-6/p}\left(\log^2 n\right)^{2/p}\log n\right)\right)$$

Attack on AMS On the negative side, we demonstrate that the classic Alon-Matias-Szegedy sketch (AMS sketch) [3], the first and perhaps most well-known F_2 estimation algorithm (which uses sub-polynomial space), is *not* adversarially robust. Specifically, we demonstrate an adversary which, when run against the AMS sketch, fools the sketch into outputting a value which is not a $(1 \pm \varepsilon)$ estimate of the F_2. The non-robustness of standard static streaming algorithms, even under simple attacks, is a further motivation to design adversarially robust algorithms.

In what follows, recall that the AMS sketch computes $S \cdot f$ throughout the stream, where $S \in \mathbb{R}^{t \times n}$ is a matrix of uniform $\{t^{-1/2}, -t^{-1/2}\}$ random variables. The estimate of the F_2 is then the value $\|Sf\|_2^2$.

Theorem 1.8 (Attack on AMS sketch; see Theorem 9.1)**.** *Let $S \in \mathbb{R}^{t \times n}$ be the AMS sketch, $1 \leq t \leq n/c$ for some constant $c > 1$. Then there is an adversary which, with probability $99/100$, succeeds in forcing the estimate $\|Sf\|_2^2$ of the AMS sketch to not be a $(1 \pm 1/2)$ approximation of the true norm $\|f\|_2^2$. Moreover, the adversary needs to only make $O(t)$ stream updates before this occurs.*

Heavy Hitters We also show how our techniques can be used to solve the popular *heavy-hitters* problem. Recall that the heavy-hitters problem tasks the streaming algorithm with returning a set S containing all coordinates i such that $|f_i| \geq \tau$, and containing no coordinates j such that $|f_j| < \tau/2$, for some threshold τ. Generally, the threshold τ is set to $\tau = \varepsilon\|f\|_p$, which is known as the L_p heavy hitters guarantee.

For L_1 heavy hitters in insertion-only streams, a deterministic $O(\frac{1}{\varepsilon}\log n)$ space algorithm exists [32]. However, for $p > 1$, specifically for the highly popular $p = 2$, things become more complicated. Note that since we can have $\|f\|_2 \ll \|f\|_1$, the L_2 guarantee is substantially stronger. For sketching-based turnstile algorithms, an $\Omega(n)$ lower bound for deterministic algorithms was previously known [13]. Since $\|f\|_1 \leq \sqrt{n}\|f\|_2$, by setting $\varepsilon = n^{-1/2}$, one can obtain a deterministic $O(\sqrt{n}\log n)$ space insertion only L_2 heavy hitters algorithm. Recently, a lower bound of $\Omega(\sqrt{n})$ for deterministic insertion only algorithms was given, demonstrating the near tightness of this result [26]. Thus, to develop a more efficient adversarially robust L_2 heavy hitters algorithm, we must employ randomness.

Indeed, by utilizing our sketch switching techniques, we demonstrate an adversarially robust L_2 heavy hitters (tracking) algorithm which uses only an $O(\varepsilon^{-1}\log\varepsilon^{-1})$ factor more space than the best known static L_2 heavy hitters tracking algorithm [8].

Theorem 1.9 (Robust L_2 heavy hitters: see Theorem 6.5)**.** *Fix any $\varepsilon > 0$. There is a streaming algorithm in the adversarial insertion only model which solves the L_2 heavy hitters problem at every step $t \in [m]$ with probability $1 - n^{-C}$ (for any constant $C > 1$). The algorithm uses $O(\frac{\log\varepsilon^{-1}}{\varepsilon^3}\log^2 n)$ bits of space.*

Entropy Estimation Additionally, we demonstrate how our sketch switching techniques can be used to obtain robust algorithms for *empirical Shannon Entropy estimation*. Here, the Shannon Entropy $H(f)$ of the stream is defined via $H(f) = -\sum_i \frac{|f_i|}{\|f\|_1}\log\left(\frac{|f_i|}{\|f\|_1}\right)$. Our results follow from an analysis of the exponential of α-Renyi Entropy, which closely approximates the Shannon entropy, showing that the former cannot rapidly change too often within the stream. Our result is an adversarially robust algorithm with space complexity only a small polylogarithmic factor larger than the best known static algorithms [11, 23].

Theorem 1.10 (Robust Entropy Estimation; see Theorem 7.3)**.** *There is an algorithm for ε-additive approximation of the Shannon entropy in the insertion-only adversarial streaming model using $O(\frac{1}{\varepsilon^5}\log^4 n)$-bits of space in the random oracle model, and $O(\frac{1}{\varepsilon^5}\log^6 n)$-bits of space in the general insertion only model.*

We remark that by making the same cryptographic assumption as in Theorem 1.3, we can remove the random oracle assumption in [23] for correctness of the entropy algorithm in the static case. Then, by applying the same techniques which resulted in Theorem 1.10, we can obtain the same stated bound for entropy with a cryptographic assumption instead of a random oracle assumption.

Bounded Deletion Streams Lastly, we show that our techniques for F_p moment estimation can be extended to data streams with a bounded number of deletions (negative updates). Specifically, we consider the bounded deletion model of [22]. Formally, given some $\alpha \geq 1$, the model enforces the restriction that at all points $t \in [m]$ in the stream, we have $\|f^{(t)}\|_p^p \geq \frac{1}{\alpha}\|h^{(t)}\|_p^p$,

where h is the frequency vector of the stream with updates $u'_i = (a_i, \Delta'_i)$ where $\Delta'_i = |\Delta_i|$ (i.e., the absolute value stream). In other words, the stream does not delete off an arbitrary amount of the F_p weight that it adds over the course of the stream.

We demonstrate that bounded deletion streams have the desirable property of having a small *flip number*, which is a measurement of how often the F_p can change substantially (see Section 3 for a formal definition). Using this property and our sketch switching technique, we obtain the following.

Theorem 1.11. *Fix $p \in [1, 2]$ and any constant $C > 1$. Then there is an adversarially robust F_p estimation algorithm which, with probability $1 - n^{-C}$, returns at each time step $t \in [m]$ an estimate R^t such that $R^t = (1 \pm \varepsilon)\|f^{(t)}\|_p^p$. The space used by the algorithm is $O(\alpha \varepsilon^{-(2+p)} \log^3 n)$.*

1.2 Other Related Work

The need for studying adversarially robust streaming and sketching algorithms has been noted before in the literature. In particular, [15, 16] motivate the adversarial model by giving applications and settings where it is impossible to assume that the queries made to a sketching algorithm are independent of the prior outputs of the algorithm, and the randomness used by the algorithm. One particularly important setting noted in [16] is when the *privacy* of the underlying data-set is a concern.

In response to this, in [20] the notion of adversarial robustness for *linear* sketching algorithms is studied. Namely, it is shown how any function $g : \mathbb{R}^n \to \mathbb{R}$, defined by $g(x) = f(Ax)$ for some $A \in \mathbb{R}^{k \times n}$ and arbitrary $f : \mathbb{R}^k \to \mathbb{R}$ cannot approximate the F_2 moment $\|x\|_2^2$ of its input to an arbitrary polynomial factor in the presence of an adversary who is allowed to query $g(x_i)$ at polynomial many points (unless k is large). Since one can insert and delete off each x_i in a turnstile stream, this demonstrates a strong lower bound for adversarially robust turnstile linear sketching algorithms.

We remark that other work has observed the danger inherent in allowing adversarial queries to a randomized sketch with only a static guarantee [1, 2]. However, the motivation of these works is slightly different, and their setting not fully adversarial. Finally, in [31], adversarial robustness of sketching in a distributed, *multi-player* model is considered, which is incomparable to the centralized streaming problem considered in this work.

2 Preliminaries

For $p > 0$, the L_p norm[4] of a vector $f \in \mathbb{R}^n$ is given by $\|f\|_p = (\sum_{i=1}^n |f_i|^p)^{1/p}$. The p-th moment, denoted by F_p, is given by $F_p = L_p^p$, or $F_p = \sum_i |f_i|^p$. For $p = 0$, we define F_0 to be the number of non-zero coordinates in f, namely $F_0 = \|f\|_0 = |\{i \ : \ f_i \neq 0\}|$. Notice that this coincides with defining $0^0 = 0$ in the prior definition of F_p. The F_0 moment is also known as the number of distinct elements. For reals $a, b \in \mathbb{R}$ and $\varepsilon > 0$, we write $a = (1 \pm \varepsilon)b$ to denote the containment $a \in [(1 - \varepsilon)b, (1 + \varepsilon)b]$. Throughout, we will often assume that our error parameter $\varepsilon > 0$ is smaller than some absolute constant ε_0 which does not depend on any of the other parameters of the problem.

A stream of length m over a domain $[n]$ is a sequence of updates $(a_1, \Delta_1), (a_2, \Delta_2) \ldots, (a_m, \Delta_m)$ where $a_t \in [n]$ an $\Delta_t \in \mathbb{Z}$. The *frequency vector* $f \in \mathbb{R}^n$ of the stream is the vector with i^{th} coordinate $f_i = \sum_{t : a_t = i} \Delta_t$. Let $f^{(t)}$ be the frequency vector restricted to the first j updates, namely $f_i^{(j)} = \sum_{t \leq j : a_t = i} \Delta_t$. It is assumed at all intermediate points $t \in [m]$ in the stream that $\|f^{(t)}\|_\infty \leq M$, and $\log(mM) = \Theta(\log n)$. Notice in particular that this bounds $|\Delta_t| \leq 2M$ for each t.

[4]Note that this is only truly a norm for $p \geq 1$.

The general model as defined above is known as the *turnstile* model of streaming. Another commonly studied model of streaming is the *insertion-only* model, where it is assumed that $\Delta_t > 0$ for each $t = 1, \ldots, m$. The insertion-only model is often presented with the following equivalent and simplified definition: an insertion only stream is given by a sequence $a_1, a_2, \ldots, a_m \in [n]$, and the frequency vector $f \in \mathbb{R}^n$ is given by the coordinates and $f_i = |\{j \in [m] : a_j = i\}|$. Since we will sometimes consider data streams with deletions (negative updates), in this work will use the former definition, where updates are pairs $(a_t, \Delta_t) \in [n] \times \mathbb{Z}$. In this paper, the space of a streaming algorithm is measured in bits, and the update time of a streaming algorithm is measured in the RAM model, where arithmetic operations on $O(\log n)$-bit integers can be done in $O(1)$ time.

The *random-oracle* model of streaming is the model where the streaming algorithm is allowed random (read-only) access to an arbitrarily long string of random bits. In other words, the space complexity of the algorithm is not charged for storing random bits. We remark that while nearly all lower bounds for streaming algorithms hold even in the random oracle model, most of our results (except for one of our results for entropy estimation and part of our cryptographic results) do not require a random oracle.

Finally, given a vector $x \in \mathbb{R}^n$, the *empirical Shannon Entropy* $H(x)$ is defined via $H(x) = -\sum_i |x_i|/\|x\|_1 \log(|x_i|/\|x\|_1)$. For $\alpha > 0$, the α-Renyi Entropy $H_\alpha(x)$ of x is given by the value $H_\alpha(x) = \log(\|x\|_\alpha^\alpha / \|x\|_1^\alpha)/(1-\alpha)$.

2.1 Tracking Algorithms

The robust streaming algorithms we design in this paper satisfy the *tracking* guarantee. Namely, they must output a response to a query at every step in time $t \in [m]$. For the case of estimation queries, this tracking guarantee is known as strong tracking.

Definition 2.1 (Strong tracking)**.** Let $f^{(1)}, f^{(2)}, \ldots, f^{(m)}$ be the frequency vectors of a stream $(a_1, \Delta_1), \ldots, (a_m, \Delta_m)$, and let $g : \mathbb{R}^n \to \mathbb{R}$ be a function on frequency vectors. A randomized algorithm $\mathcal{A}$ is said to provide (ε, δ)-*strong* g-*tracking* if at each time step $t \in [m]$ it outputs an estimate R_t such that

$$|R_t - g(f^{(t)})| \leq \varepsilon |g(f^{(t)})|$$

for all $t \in [m]$ with probability at least $1 - \delta$.

In contrast, *weak tracking* replaces the error term $\varepsilon|g(f^{(t)})|$ by $\max_{t' \in [m]} \varepsilon \cdot |g(f^{(t')})|$. However, for the purposes of this paper, we will not need to consider weak tracking. We now state two results for strong tracking of F_p moments for $p \in [0, 2]$. Both results are for the static setting, i.e., for a stream fixed in advance (and not for the adaptive adversarial setting that we consider).

Lemma 2.2 ([7])**.** *For $0 < p \leq 2$, there is an insertion only streaming algorithm which provides (ε, δ)-strong F_p-tracking using $O(\frac{\log n}{\varepsilon^2}(\log \varepsilon^{-1} + \log \delta^{-1} + \log\log n))$ bits of space.*

Lemma 2.3 ([6])**.** *There is an insertion-only streaming algorithm which provides (ε, δ)-strong F_0-tracking using $O(\frac{\log\log n + \log \delta^{-1}}{\varepsilon^2} + \log n)$ bits of space.*

2.2 Roadmap

In Section 3, we introduce our two general techniques for transforming static streaming algorithms into adversarially robust algorithms. In Section 4, we give our results on estimation of F_p moments, and in Section 5 we give our algorithms for adversarially robust distinct elements estimation. Next, in Section 6, we introduce our robust L_2 heavy hitters algorithm, and in Section 7 we give our entropy estimation algorithm. In Section 8, we provide our algorithms for F_p moment estimation in the bounded deletion model. In Section 9, we give our adversarial attack on the AMS sketch. Finally, in Section 10, we give our algorithm for optimal space distinct elements estimation under cryptographic assumptions.

3 Tools for Robustness

In this section we establish two methods, *sketch switching* and *computation paths*, allowing one to convert an approximation algorithm for any sufficiently well-behaved streaming problem to an adversarially robust one for the same problem. At the core of these methods is a rounding technique that allows us to leak only a small amount of information to the adversary. The relevant definitions, of rounding and flip number, are given first, and afterward we use them to develop our robustification methods.

For any $x \in \mathbb{R}$ and $\varepsilon > 0$, define the real number $[x]_\varepsilon$ as follows. If $x > 0$, then $[x]_\varepsilon$ is the number $y > 0$ of the form $y = (1+\varepsilon)^\ell$ for $\ell \in \mathbb{Z}$ which minimizes $\max\{y/x, x/y\}$. In other words, $[x]_\varepsilon$ is the power of $(1+\varepsilon)$ closest (in multiplicative terms) to x. If $x < 0$, set $[x]_\varepsilon$ as $-[-x]_\varepsilon$; and finally, set $[0]_\varepsilon = 0$. Note that $[x]_\varepsilon$ is in all cases a $(1+\varepsilon/2)$-multiplicative approximation of x. Furthermore, in this paper the values of x that we consider are generally in the range $[-n^c, -1/n^c] \cup \{0\} \cup [1/n^c, n^c]$. The number of possible values of $[x]_\varepsilon$ for x in this range is $O(\varepsilon^{-1} \log n)$ and thus no more than $O(\log\log n + \log 1/\varepsilon)$ bits are required to store $[x]_\varepsilon$.

Definition 3.1 (ε-rounding)**.** Let $y_1, y_2, \ldots, y_m \in \mathbb{R}$ be a sequence of non-negative real numbers. An *ε-rounding* of $y_1, \ldots, y_m$ is a sequence $y'_1, \ldots, y'_m$ constructed in the following way. We set $y'_1 = [y_1]_\varepsilon$. Given y'_i, we construct y'_{i+1} by setting $y'_{i+1} = y'_i$ if $(1-\varepsilon)y_{i+1} \le y'_i \le (1+\varepsilon)y_{i+1}$, otherwise we set $y'_{i+1} = [y_{i+1}]_\varepsilon$.

The notion of ε-rounding is useful as it allows a streaming algorithm to leak less information, without a significant loss in accuracy; specifically, this can be done by outputting the ε-rounding of the output of the algorithm, instead of the raw output.

Typical streaming problems ask one to $(1+\varepsilon)$-approximate the value of some function $g : \mathbb{R}^n \to \mathbb{R}$ of the frequency vector. For example, in the count-distinct problem, $g(f^{(i)})$ is the number of distinct elements among $(a_1, \Delta_1), \ldots, (a_i, \Delta_i)$. As it turns out, for many classical problems in the insertion-only model (including count-distinct), the number of distinct values among the ε-rounding of the sequence $g(f^{(1)}), g(f^{(2)}), \ldots, g(f^{(m)})$ is small. The next definition, of a flip number, captures a closely related quantity.

Definition 3.2 ((ε, m)-flip number)**.** Let $\varepsilon > 0$ and $m \in \mathbb{N}$, and let $y_0, y_1, \ldots, y_m$ be any sequence of real numbers. The *(ε, m)-flip number* of $y_0, y_1, \ldots, y_m$ is the maximum $k \in \mathbb{N}$ for which there exist $0 \le i_1 < \ldots < i_k \le m$ so that $y_{i_{j-1}} \notin [(1-\varepsilon)y_{i_j}, (1+\varepsilon)y_{i_j}]$ for every $j = 2, 3, \ldots, k$.

Given a function $g\colon \mathbb{R}^n \to \mathbb{R}$, the (ε, m)-flip number $\lambda_{\varepsilon,m}(g)$ of g is the maximum, over all sequences $(a_1, \Delta_1), \ldots, (a_m, \Delta_m)$ of possible stream updates, of the (ε, m)-flip number of the sequence $y_0, y_1, \ldots, y_m$ defined by $y_i = g(f^{(i)})$ for any $0 \le i \le m$, where as usual $f^{(i)}$ is the frequency vector after stream updates $(a_1, \Delta_1), \ldots, (a_i, \Delta_i)$ (and $f^{(0)}$ is the n-dimensional zeros vector).

Note that flip number is clearly monotone in ε: namely $\lambda_{\varepsilon,m}(g) \le \lambda_{\varepsilon',m}(g)$ if $\varepsilon' < \varepsilon$. For our purposes, we will need a robust variant of this property, requiring that any sequence that approximates the elements of $(g(f^{(i)}))_{i=1}^m$ to within an error of, say, $\varepsilon/10$, will have its ε-rounding not change often.

Lemma 3.3. *Let $g\colon \mathbb{R}^n \to \mathbb{R}$, $0 < \varepsilon < c$ where c is a small positive absolute constant, and $m \in \mathbb{N}$. Suppose that $z_0, z_1, \ldots, z_m$ is a sequence of real numbers satisfying the following: there exists a stream of updates $(a_1, \Delta_1), \ldots, (a_m, \Delta_m)$ on which the output of g is $y_0, y_1, \ldots, y_m$, where $(1-\varepsilon/10)y_i \le z_i \le (1+\varepsilon/10)y_i$. Let $z'_0, z'_1, \ldots, z'_m$ be the ε-rounding of $z_0, z_1, \ldots, z_m$. Then the number of indices $i \in [m]$ for which $z'_i \ne z'_{i-1}$ is at most $\lambda_{\varepsilon/10,m}(g)$.*

Proof. Consider the collection of all indices $i \in [m]$ for which $z'_i \ne z'_{i-1}$. Write these indices as $1 \le i_1 < i_2 < \ldots < i_k \le m$. We wish to show that $|y_{i_j} - y_{i_{j-1}}| > \varepsilon|y_{i_j}|/10$ for any $2 \le j \le k$, which will immediately imply by definition that $\lambda_{\varepsilon/10,m}(g) \ge k$.

Indeed, observe that all of the following hold for any $2 \leq j \leq k$.

- By definition of ε-rounding, we know that $z'_{i_{j-1}} \notin [(1-\varepsilon)z_{i_j}, (1+\varepsilon)z_{i_j}]$.
- Again by definition of rounding, $z'_{i_{j-1}}$ is the power of $1+\varepsilon$ closest to $z_{i_{j-1}}$, and so $z'_{i_{j-1}} = z_{i_{j-1}}(1 \pm \varepsilon/2)$.
- By the assumption in the statement of the lemma, $z_i = y_i(1 \pm \varepsilon/10)$ for both $i = i_{j-1}$ and $i = i_j$.

It follows that the difference $|y_{i_j} - y_{i_{j-1}}|$ is at least of the form[5] $(3\varepsilon/10 - O(\varepsilon^2))|y_{i_j}| > \varepsilon|y_{i_j}|/10$, where the inequality holds provided that $\varepsilon < c$ for some absolute constant $c > 0$. ■

Note that the flip number of a function critically depends on the model in which we work, as the maximum is taken over all sequences of *possible stream updates*; for insertion-only streams, the set of all such sequences is more limited than in the general turnstile model, and correspondingly many streaming problems have much smaller flip number when restricted to the insertion only model. We now give an example of a class of functions with bounded flip number.

Proposition 3.4. *Let $g : \mathbb{R}^n \to \mathbb{R}$ be any monotone function, meaning that $g(x) \geq g(y)$ if $x_i \geq y_i$ for each $i \in [n]$. Assume further that $g(0) = 0$, $g(x) \geq T^{-1}$ for all $x > 0$, and $g(M \cdot \vec{1}) \leq T$, where M is the bound on the entries of the frequency vector and $\vec{1}$ is the all 1's vector. Then the flip number of g in the insertion only streaming model is $\lambda_{\varepsilon,m}(g) = O(\frac{1}{\varepsilon}\log T)$.*

Proof. To see this, note that $g(f^{(0)}) = 0$, $g(f^{(1)}) \geq T^{-1}$, and $g(f^{(m)}) \leq g(\vec{1} \cdot M) \leq T$. Since the stream has only positive updates, $g(f^{(0)}) \leq g(f^{(1)}) \leq \cdots \leq g(f^{(m)})$. Let $y_1, \ldots, y_k \in [m]$ be any set of points such that $g(f^{(y_i)}) < (1+\varepsilon)g(f^{(y_{i+1})})$ for each i. Since there are at most $O(\frac{1}{\varepsilon}\log T)$ powers of $(1+\varepsilon)$ between T^{-1} and T, by the pigeonhole principle if $k > \frac{C}{\varepsilon}\log(T)$ for a sufficiently large constant C, then at least two values must satisfy $(1+\varepsilon)^j \leq g(f^{(y_i)}) \leq g(f^{(y_{i+1})}) \leq (1+\varepsilon)^{j+1}$ for some j, which is a contradiction. ■

Note that a special case of the above are the F_p moments of a data stream. Recall here $\|x\|_0 = |\{i : x_i \neq 0\}|$ is the number of non-zero elements in a vector x.

Corollary 3.5. *Let $p > 0$. Then the (ε, m)-flip number of $\|x\|_p^p$ in the insertion only streaming model is $\lambda_{\varepsilon,m}(\|\cdot\|_p^p) = O(\frac{1}{\varepsilon}\log m)$ for $p \leq 2$, and $\lambda_{\varepsilon,m}(\|\cdot\|_p^p) = O(\frac{p}{\varepsilon}\log m)$ for $p > 2$. For $p = 0$, we also have $\lambda_{\varepsilon,m}(\|\cdot\|_0) = O(\frac{1}{\varepsilon}\log m)$*

Proof. We have $\|\vec{0}\|_p^p = 0$, $\|z\|_p^p \geq 1$ for any non-zero $z \in \mathbb{Z}$, and $\|f^{(m)}\|_p^p \leq M^p n \leq n^{cp}$ for some constant c, where the second to last inequality holds because $\|f\|_\infty \leq M$ for some $M = \text{poly}(n)$ is assumed at all points in the streaming model. Moreover, for $p = 0$ we have $\|f^{(m)}\|_0 \leq n$. The result then follows from applying Proposition 3.4 with $T = n^{c \cdot \max\{p,1\}}$. ■

Another special case of Proposition 3.4 concerns the *Cascaded Norms* of insertion-only data streams [24]. Here, the frequency vector f is replaced with a matrix $A \in \mathbb{Z}^{n \times d}$, which receives coordinate-wise updates in the same fashion, and the (p,k) cascaded norm of A is given by $\|A\|_{(p,k)} = (\sum_i (\sum_j |A_{i,j}|^k)^{p/k})^{1/p}$. In other words, $\|A\|_{(p,k)}$ is the result of first taking the L_k norm of the rows of A, and then taking the L_p norm of the result. Proposition 3.4 similarly holds with $T = \text{poly}(n)$ in the insertion only model, and therefore the black-box reduction techniques introduced in the following sections are also applicable to these norms (using e.g., the cascaded algorithms of [24]).

Having a small flip number is very useful for robustness, as our next two robustification techniques demonstrate.

[5] Here we think of $\varepsilon > 0$ as a small parameter.

3.1 The Sketch Switching Technique

Our first technique is called *sketch switching*, and is described in Algorithm 1. The technique maintains multiple instances of a static strong tracking algorithm, where each time step only one of the instances is "active". The idea is to change the current output of the algorithm very rarely. Specifically, as long as the current output is a good enough multiplicative approximation of the estimate of the active instance, the estimate we give to the adversary does not change, and the current instance remains active. As soon as this approximation guarantee is not satisfied, we update the output given to the adversary, deactivate our current instance, and activate the next one in line. By carefully exposing the randomness of our multiple instances, we show that the strong tracking guarantee (which a priori holds only in the static setting) can be carried into the robust setting. The number of instances required is controlled by the flip number of the problem.

Algorithm 1: Adversarially Robust g-estimation by Sketch Switching

$\lambda \leftarrow \lambda_{\varepsilon/20,m}(g)$
Initialize independent instances $A_1, \ldots, A_\lambda$ of a $(\varepsilon/20, \delta/\lambda)$-strong g-tracking algorithm.
$\rho \leftarrow 1$
$\tilde{g} \leftarrow g(\vec{0})$
while *new stream update* (a_k, Δ_k) **do**
    Insert update (a_k, Δ_k) into each algorithm $A_1, \ldots, A_\lambda$.
    $y \leftarrow$ current output of A_ρ.
    if $\tilde{g} \notin (1 \pm \frac{\varepsilon}{2})y$ **then**
        $\tilde{g} \leftarrow [A_\rho(k)]_{\varepsilon/2}$.
        $\rho \leftarrow \rho + 1$.
    Output estimate $\tilde{g}$.
end

Lemma 3.6 (Sketch Switching)**.** *Fix any function $g : \mathbb{R}^n \to \mathbb{R}$ and let A be a streaming algorithm that for any $\varepsilon, \delta > 0$ uses space $L(\varepsilon, \delta)$, and satisfies the (ε, δ)-strong g-tracking property on the frequency vectors $f^{(1)}, \ldots, f^{(m)}$ of any particular fixed stream. Then Algorithm 1 is an adversarially robust algorithm for $(1+\varepsilon)$-approximating $g(f^{(t)})$ at every step $t \in [m]$ with success probability $1 - \delta$, whose space is $O\left(L(\varepsilon_0, \delta_0) \cdot \lambda_{\varepsilon_0,m}(g)\right)$, where $\varepsilon_0 = \Theta(\varepsilon), \delta_0 = \Theta(\delta/\lambda_{\varepsilon_0,m}(g))$.*

Proof. Note that for a fixed randomized algorithm $\mathcal{A}$, we can assume the adversary against $\mathcal{A}$ is deterministic without loss of generality. This is because given a randomized adversary and algorithm, if the adversary succeeds with probability greater than δ in fooling the algorithm, then by a simple averaging argument there must exist a fixing of the random bits of the adversary which fools $\mathcal{A}$ with probability greater than δ over the coin flips of $\mathcal{A}$. Note also here that conditioned on a fixing of the randomness for both the algorithm and adversary, the entire stream and behavior of both parties is fixed. We thus start by fixing such a string of randomness for the adversary, which makes it deterministic. As a result, suppose that y_i is the output of the streaming algorithm on step i. Then given $y_1, y_2, \ldots, y_k$ and the stream updates $(a_1, \Delta_1), \ldots, (a_k, \Delta_k)$ so far, the next stream update (a_{k+1}, Δ_{k+1}) is deterministically fixed. We stress that the randomness of the algorithm is not fixed at this point; we will gradually reveal it along the proof.

We may assume that $\varepsilon > 0$ is small enough, in particular smaller than the positive absolute constant dictated by Lemma 3.3. Now let $\varepsilon_0 = \varepsilon/20$ and $\lambda = \lambda_{\varepsilon_0,m}$, and fix a set of λ independent instances $A_1, A_2, \ldots, A_\lambda$ of the streaming algorithm A with the $(\varepsilon_0, \delta_0)$-strong g-tracking property. Since $\delta_0 = O(\delta/\lambda)$, later on we will be able to union bound over the assumption

that for all $\rho \in [\lambda]$, A_i satisfies strong tracking on some fixed stream (to be revealed along the proof); the stream corresponding to A_ρ will generally be different than that corresponding to ρ' for $\rho \neq \rho'$.

First, let us fix the randomness of the first instance, A_1. Let $u_1^1, u_2^1, \ldots, u_m^1$ be the updates $u_j^1 = (a_j, \Delta_j)$ that the adversary would make if $\mathcal{A}$ were to output $y_0 = g(\vec{0})$ at every time step, and let $f^{(t),1}$ be the stream vector after updates $u_1^1, \ldots, u_t^1$. Let $A_1(t)$ be the output of algorithm A_1 at time t of the stream $u_1^1, u_2^1, \ldots, u_t^1$. Let $t_1 \in [m]$ be the first time step such that $|A_1(t_1) - y_0| > \varepsilon |A_1(t_1)|/2$, if it (if not we can set, say, $t_1 = m + 1$). At this point, we change our output to $y_1 = [A_1(t_1)]_{\varepsilon/2}$. Assuming that A_1 satisfies strong tracking for g with approximation parameter ε_0 with respect to the fixed stream of updates $u_1^1, \ldots, u_m^1$, we know that $A_1(t) = (1 \pm \varepsilon_0) g(f^{(t)})$ for each $t < t_1$ and that $y_0 = (1 \pm \varepsilon/2)|A_1(t_1)|$, from which we conclude that $y_0 = (1 \pm \varepsilon) g(f^{(t)})$ for any $t < t_1$. Similarly, since $y_1 = [A_1(t_1)]_{\varepsilon/2}$ we know that $y_1 = (1 \pm /\varepsilon/2) A_1(t_1)$ and so $y_1 = (1 \pm \varepsilon) g(f^{(t_1)})$ holds as well, that is, the new output of our algorithm y_1 is still a valid approximation at time $t = t_1$.

At this point, the algorithm "switches" to the instance A_2, and presents y_1 as its output as long as $A_2(t) = (1 \pm \varepsilon/2) y_1$. Recall that randomness of the adversary is already fixed, and consider the sequence of updates obtained by concatenating $u_1^1, \ldots, u_{t_1}^1$ as defined above (these are the updates already sent by the adversary) with the sequence $u_{t_1+1}^2, \ldots, u_m^2$ to be sent by the adversary if the output from time $t = t_1$ onwards would always be y_1. We condition on the ε_0-strong g-tracking guarantee on A_2 holding for this fixed sequence of updates, noting that this is the point where the randomness of A_2 is revealed. Set $t = t_2$ as the first value of t (if exists) for which $A_2(t) = (1 \pm \varepsilon/2) y_1$ does not hold. We now have, similarly to above, $y_1 = (1 \pm \varepsilon) g(f^{(t)})$ for any $t_1 \leq t < t_2$, and $y_2 = (1 \pm \varepsilon) g(f^{(t_2)})$.

The same reasoning can be applied inductively for A_ρ, for any $\rho \in [\lambda_{\varepsilon_0, m}]$, to get that the output of our algorithm y_ρ is within a $(1 \pm \varepsilon)$-multiplicative factor for any of the time steps $t = t_\rho, t_\rho + 1, \ldots, \min\{t_{\rho+1} - 1, m\}$. It remains to verify that this strategy will succeed in handling all m elements of the stream (and will not exhaust its pool of algorithm instances before then). Indeed, this follows immediately from Lemma 3.3, applied with the parameters $g, \varepsilon/2, m$; since our instances of the algorithm employ $(\varepsilon/20)$-strong tracking for g, and our actual output is an $(\varepsilon/2)$-rounding of these instances, the conditions of Lemma 3.3 hold as required. ■

3.2 The Bounded Computation Paths Technique

With our sketch switching technique, we showed that maintaining multiple instances of a non-robust algorithm to estimate a function g, and switching between them when the rounded output changes, is a recipe for a robust algorithm to estimate g. We next provide another recipe, which keeps only one instance, whose success probability for any fixed stream is very high; it relies on the fact that if the flip number is small, then the total number of fixed streams that we should need to handle is also relatively small, and we will be able to union bound over all of them. Specifically, we show that any non-robust algorithm for a function with bounded flip number can be modified into an adversarially robust one by setting the failure probability δ small enough.

Definition 3.7 (ε-rounding for algorithms)**.** The ε-rounding of a (possibly randomized) streaming algorithm A is an algorithm A' which simulates an instance of A and runs as follows: after the first received element, A' return $z_1' = [z_1]_\varepsilon$, where z_1 is the output of A. When the i-th stream element is received, for $i \geq 2$, the algorithm A' feeds this element to its instance of A and receives an output z_i. if $(1 - \varepsilon) z_i \leq z_{i-1}' \leq (1 + \varepsilon) z_i$ then the algorithm sets $z_i' = z_{i-1}'$. Otherwise, it sets $z_i' = [z_i]_\varepsilon$. In both cases, the output of A' is z_i'.

Note that for any fixed stream of updates $(a_1, \Delta_1), \ldots, (a_m, \Delta_m)$, the output sequence $z_1', \ldots, z_m'$ of the ε-rounding algorithm A' is simply the ε-rounding for the output sequence $z_1, \ldots, z_m$ of its internally maintained A-instance.

Lemma 3.8 (Computation Paths)**.** *Fix $g\colon \mathbb{R}^n \to \mathbb{R}$ and suppose that the output of g over any possible stream of updates is in the range $[-T,-1/T]\cup\{0\}\cup[1/T,T]$ for some $T>1$. Let A be a streaming algorithm that for any $\varepsilon,\delta>0$ satisfies the (ε,δ)-strong g-tracking property on the frequency vectors $f^{(1)},\dots,f^{(m)}$ of any particular fixed stream. Then the ε_0-rounding algorithm A' for A, where A is instantiated with approximation parameter $\varepsilon_0=\Theta(\varepsilon)$ and failure probability $\delta_0=\delta/\left(\binom{m}{\lambda_{\varepsilon_0,m}(g)}\cdot\left(\Theta(\varepsilon^{-1}\log T)\right)^{\lambda_{\varepsilon_0,m}(g)}\right)$, is an adversarially robust algorithm for $(1+\varepsilon)$-approximating $g(f^{(t)})$ in all steps $t\in[m]$, with success probability $1-\delta$.*

Proof. As in the proof of Lemma 3.6, we may assume the adversary to be deterministic. This means, in particular, that the output sequence we provide to the adversary fully determines its stream of updates $(a_1,\Delta_1),\dots,(a_m,\Delta_m)$. Take $\varepsilon_0=\varepsilon/20$ and take $\lambda=\lambda_{\varepsilon_0,m}(g)$. Now consider all sequences $s_0,s_1,\dots,s_m$ where each s_i is either zero, or a power of $1+\varepsilon$ between $1/T(1+\varepsilon)$ and $T(1+\varepsilon)$, or the negation of such a number; and moreover, the number of $i\in[m]$ for which $s_i\neq s_{i-1}$ is at most λ. By a standard counting argument, the number of such sequences is $\binom{m}{\lambda_{\varepsilon_0,m}(g)}\cdot\left(O(\varepsilon^{-1}\log T)\right)^{\lambda_{\varepsilon_0,m}(g)}$. Each such output sequence uniquely determines a corresponding stream of updates for the deterministic adversary; let $\mathcal{S}$ be the collection of all such streams.

Pick $\delta_0=\delta/|\mathcal{S}|$. Taking a union bound, we conclude that with probability $1-\delta$, A (instantiated with parameters ε_0 and δ_0) provides an ε_0-strong g-tracking guarantee for all streams in $\mathcal{S}$. We now fix the randomness of A, and assume the last event holds. For any stream of updates $((a_1,\Delta_1),\dots,(a_m,\Delta_m))\in\mathcal{S}$ this fixes a corresponding output sequence $(z_0,\dots,z_m)$ of the algorithm A. Let $\mathcal{O}$ be the collection of all such output sequences. Each such sequence satisfies the conditions of Lemma 3.3, and so its $\varepsilon/2$-rounding is a sequence $z_0',\dots,z_m'$ where $z_i'\neq z_{i-1}'$ for at most λ values of $i\in[m]$. Thus, if the above event holds, then the output provided to the adversary has at most λ distinct values (which are powers of $1+\varepsilon$ in the relevant range). Thus, conditioning on this event, the stream of updates which the adversary creates upon observing the rounded outputs $(z_0',\dots,z_m')$ of the algorithm A' must by definition be in $\mathcal{S}$. It follows by the ε_0-strong tracking guarantee given by this event that the $\varepsilon/2$-rounding $z_0',\dots,z_m'$ of the output of A is a $(1\pm\varepsilon)$-approximation for g on all possible adversarial streams in this setting, i.e., all streams in $\mathcal{S}$. ∎

4 F_p Estimation

In this section, we introduce our adversarially robust F_p moment estimation algorithms. Recall that F_p is given by $\|f\|_p^p=\sum_i|f_i|^p$ for $\mathfrak{p}>0$. For $p=0$, the F_0 moment, or the number of distinct elements, is the number of non-zero coordinates in f, that is, $\|f\|_0=|\{i\in[n]:f_i\neq 0\}|$. Recall that in Corollary 3.5, we bounded the flip number of the F_p moment in insertion only streams for any fixed $p>0$ by $O(p\varepsilon^{-1}\log n)$. By using our sketch switching argument, the strong F_p tracking guarantees of [7] as stated in Lemma 2.2, we obtain our first result for $0<p\leq 2$.

Theorem 4.1 (F_p estimation by sketch switching)**.** *Fix any $0<\varepsilon,\delta\leq 1$ and $0<p\leq 2$. There is a streaming algorithm for the insertion-only adversarial model which, with probability $1-\delta$, successfully outputs at each step $t\in[m]$ a value R^t such that $R^t=(1\pm\varepsilon)\|f^{(t)}\|_p$. The space used by the algorithm is*

$$O\left(\frac{1}{\varepsilon^3}\log n\log\varepsilon^{-1}(\log\varepsilon^{-1}+\log\delta^{-1}+\log\log n)\right)$$

Proof. By an application of Lemma 3.6 along with the flip number bound of Corollary 3.5 and the strong tracking algorithm of Lemma 2.2, we immediately obtain a space complexity of

$$O\left(\frac{1}{\varepsilon^3}\log^2 n(\log\varepsilon^{-1}+\log\delta^{-1}+\log\log n)\right)$$

We now describe how the factor of $\frac{1}{\varepsilon}\log n$, coming from running $\lambda_{\varepsilon,m} = \Theta(\frac{1}{\varepsilon}\log n)$ independent sketches in Lemma 3.6, can be improved to $\frac{1}{\varepsilon}\log \varepsilon^{-1}$.

To see this, we change Algorithm 1 in the following way. Instead of $\Theta(\frac{1}{\varepsilon}\log n)$ independent sketches, we use $\lambda \leftarrow \Theta(\frac{1}{\varepsilon}\log \varepsilon^{-1})$ independent sketches, and change line 10 to state $\rho \leftarrow \rho + 1 \pmod{\lambda}$. Each time we change ρ to $\rho + 1$ and begin using the new sketch $A_{\rho+1}$, we *completely restart* the algorithm A_ρ with new randomness, and run it on the remainder of the stream. The proof of correctness in Lemma 3.6 is completely unchanged, except for the fact that now A_ρ is run only on a *suffix* $a_j, a_{j+1}, \ldots,$ of the stream, if j is the time step where A_ρ is reinitialized. Specifically, at each time step $t \geq j$, A_ρ will produce a $(1 \pm \varepsilon)$ estimate of $\|f^{(t)} - f^{(j-1)}\|_p$ instead of $\|f^{(t)}\|_p$. However, since the sketch will not be used again until a time step t' where $\|f^{(t)}\|_p \geq (1+\varepsilon)^\lambda \|f^{(j)}\|_p = \frac{100}{\varepsilon}\|f^{(j)}\|_p$, it follows that only an ε fraction of the ℓ_p mass was missed by A_ρ. In particular, $\|f^{(t')} - f^{(j-1)}\|_p = (1 \pm \varepsilon/100)\|f^{(t')}\|_p$, and thus by giving a $(1 \pm \varepsilon/10)$ approximation of $\|f^{(t')} - f^{(j-1)}\|_p$, the algorithm A_ρ gives the desired $(1 \pm \varepsilon)$ approximation of the underlying ℓ_p norm, which is the desired result after a constant factor rescaling of ε. Note that this argument could be used for the L_0 norm, or any p norm for $p \geq 0$, using a F_p strong tracking algorithm for the relevant p. ■

While for most values of δ, the above theorem has better space complexity than the computation paths reduction, for the regime of very small failure probability it is actually preferable to use the latter, as we now state. The proof is a direct application of Lemma 3.8, along with the flip number bound of Corollary 3.5, and the $O(\varepsilon^{-2}\log n \log \delta^{-1})$ static F_p estimation algorithm of [27]. We remark that the below complexity is optimal up to a $\log n$ term for insertion only streams. The reason is that the $O(1/\varepsilon^2 \log n \log 1/\delta)$ lower bound of [25] degrades to $O(1/\varepsilon^2 \log 1/\delta)$ when deletions are not allowed. [6]

Theorem 4.2 (F_p estimation for small δ)**.** *Fix any $0 < \varepsilon < 1$, $0 < p \leq 2$, and $\delta < n^{-C\frac{1}{\varepsilon}\log n}$ for a sufficiently large constant $C > 1$. There is a streaming algorithm for the insertion-only adversarial model which, with probability $1 - \delta$, successfully outputs at each step $t \in [m]$ a value R^t such that $R^t = (1 \pm \varepsilon)\|f^{(t)}\|_p$. The required space is $O\left(\frac{1}{\varepsilon^2}\log n \log \delta^{-1}\right)$ bits.*

Next, we show that for turnstile streams with F_p flip number λ, we can estimate F_p with error probability $\delta = n^{-\lambda}$. The space requirement of the algorithm is optimal for algorithms with such failure probability δ, which follows by an $\Omega(\varepsilon^{-2}\log n \log \delta^{-1})$ lower bound for turnstile algorithms [25], where the hard instance in question has small F_p flip number. [7]

Theorem 4.3 (F_p Estimation for λ-flip number turnstile streams)**.** *Let $\mathcal{S}_\lambda$ be the set of all turnstile streams with F_p flip number at most $\lambda \geq \lambda_{\varepsilon,m}(\|\cdot\|_p^p)$ for any $0 < p \leq 2$. Then there is an adversarially robust streaming algorithm for the class $\mathcal{S}_\lambda$ of streams that, with probability $1 - n^{-C\lambda}$ for any constant $C > 0$, outputs at each time step a value R^t such that $R^t = (1 \pm \varepsilon)\|f\|_p^p$. The space used by the algorithm is $O(\varepsilon^{-2}\lambda \log^2 n)$.*

Proof. The proof follows by simply applying Lemma 3.8, along with the $O(\varepsilon^{-2}\log n \log \delta^{-1})$ bit turnstile algorithm of [27]. ■

In addition, we show that the F_p moment can also be robustly estimated for $p > 2$. In this case, it is preferable to use our computation paths reduction, because the upper bounds for F_p moment estimation for large p yield efficiency gains when setting δ to be small.

Theorem 4.4 (F_p estimation, $p > 2$, by Computation Paths)**.** *Fix any $\varepsilon, \delta > 0$, and any constant $p > 2$. Then there is a streaming algorithm for the insertion-only adversarial model*

[6] Specifically, one may not apply the augmented indexing step in the reduction of [25] to delete off coordinates in $\log n$ levels, which loses this factor in the lower bound.

[7] The hard instance in [25] is a stream where $O(n)$ updates are first inserted and then deleted, thus the flip number is at most twice the F_p flip number of an insertion only stream.

which, with probability $1 - n^{-(c \log n)/\varepsilon}$ *for any constant* $c > 1$*, successfully outputs at every step* $t \in [m]$ *a value* R^t *such that* $R^t = (1 \pm \varepsilon)\|f^{(t)}\|_p$*. The space used by the algorithm is* $O(n^{1-2/p}(\varepsilon^{-3} \log^2 n + \varepsilon^{-6/p}(\log^2 n)^{2/p} \log n))$*.*

Proof. We use the insertion only F_p estimation algorithm of [14], which achieves

$$\left(n^{1-2/p}\left(\varepsilon^{-2} \log \delta^{-1} + \varepsilon^{-4/p} \log^{2/p} \delta^{-1} \log n\right)\right)$$

bits of space in the turnstile (and therefore insertion only) model. We can set $\delta = \delta/m$ to union bound over all steps, making it a strong F_p tracking algorithm with

$$O\left(n^{1-2/p}\left(\varepsilon^{-2} \log(n\delta^{-1}) + \varepsilon^{-4/p} \log^{2/p}(n\delta^{-1}) \log n\right)\right)$$

bits of space. Then by Lemma 3.8 along with the flip number bound of Corollary 3.5, the claimed space complexity follows. ■

5 Distinct Elements Estimation

We now demonstrate how our sketch switching technique can be used to estimate the number of *distinct elements*, also known as F_0 estimation, in an adversarial stream. In this case, since there exist static F_0 strong tracking algorithms [6] which are more efficient than repeating the sketch $\log \delta^{-1}$ times, it will be preferable to use our sketch switching technique.

Theorem 5.1 (Robust Distinct Elements by Sketch Switching)**.** *There is an algorithm which, when run on an adversarial insertion only stream, produces at each step* $t \in [m]$ *an estimate* R^t *such that* $R^t = (1 \pm \varepsilon)\|f^{(t)}\|_0$ *with probability at least* $1 - \delta$*. The space used by the algorithm is* $O(\frac{\log \varepsilon^{-1}}{\varepsilon}(\frac{\log \varepsilon^{-1} + \log \delta^{-1} + \log \log n}{\varepsilon^2} + \log n))$ *bits.*

Proof. We use the insertion only distinct elements *strong* tracking algorithm of [6]. Specifically, the algorithm of [6] uses space $O(\frac{\log \delta_0^{-1} + \log \log n}{\varepsilon^2} + \log n)$, and with probability $1 - \delta_0$, successfully returns an estimate R^t for every step $t \in [m]$ such that $R^t = (1 \pm \varepsilon)\|f^{(t)}\|_0$ in the non-adversarial setting. Then by application of Lemma 3.6, along with the flip number bound of $O(\log n/\varepsilon)$ from Corollary 3.5, we obtain the space complexity with a factor of $\frac{\log n}{\varepsilon}$ blow-up after setting $\delta_0 = \Theta(\delta \frac{\varepsilon}{\log n})$. This gives a complexity of $O(\frac{\log n}{\varepsilon}(\frac{\log \varepsilon^{-1} + \log \delta^{-1} + \log \log n}{\varepsilon^2} + \log n))$. To reduce the extra $\log n$-factor to a $\log \varepsilon^{-1}$ factor, we just apply the same argument used in the proof of Theorem 4.1, which shows that by restarting sketches it suffices to keep only $O(\varepsilon^{-1} \log \varepsilon^{-1})$ copies. ■

5.1 Fast Distinct Elements Estimation

As noted earlier, there are many reasons why one may prefer one of the reductions from Section 3 to the other. In this section, we will see such a motivation. Specifically, we show that adversarially robust L_0 estimation can be accomplished with extremely fast update time using the computation paths reduction of Lemma 3.8.

First note that the standard approach to obtaining failure probability δ is to repeat the estimation algorithm $\log \delta^{-1}$ times independently, and take the median output. However, this blows up the update time by a factor of $\log \delta^{-1}$. Thus black-box applying Lemma 3.8 by setting δ to be small can result in a larger update time. To improve upon this, we will introduce an insertion only distinct elements estimation algorithm, with the property that the runtime dependency on δ^{-1} is extremely small (roughly $\log^2 \log \log \delta^{-1}$). Thus applying Lemma 3.8 on this algorithm results in a very fast robust streaming algorithm.

Lemma 5.2. *There is a streaming algorithm which, with probability $1-\delta$, returns a $(1\pm\varepsilon)$ multiplicative estimate of the number of distinct elements in an insertion only data stream. The space required is $O(\frac{1}{\varepsilon^2}\log n(\log\log n+\log\delta^{-1}))$,*[8] *and the worst case running time per update is $O\left(\left(\log^2\log\frac{\log n}{\delta}\right)\cdot\left(\log\log\log\frac{\log n}{\delta}\right)\right)$.*

Before stating our proof of Lemma 5.2, We will begin with the following proposition which will allow for the fast evaluation of d-wise independent hash functions.

Proposition 5.3 ([36], Ch. 10)**.** *Let R be a ring, and let $p\in R[x]$ be a degree d univariate polynomial over R. Then given distinct $x_1,x_2,\ldots,x_d\in R$, all the values $p(x_1),p(x_2),\ldots,p(x_d)$ can be computed using $O(d\log^2 d\log\log d)$ operations over R.*

Proof of Lemma 5.2. We describe the algorithm here, as stated in Algorithm 2.

Algorithm 2: Fast non-adversarial distinct elements estimation.

```
Initialize Lists L_0, L_1, ..., L_t ← ∅, for t = Θ(log n)
B ← Θ(ε^{-2} log δ^{-1}),  d ← Θ(log log n + log δ^{-1})
Initialize d-wise independent hash function H : [n] → [2^ℓ] such that n^2 ≤ 2^ℓ ≤ n^3.
while Receive update (a_t, Δ_t) ∈ [n] × Z do
    Let j be such that 2^{ℓ-j-1} ≤ H(a_t) < 2^{ℓ-j}
    if L_j has not been deleted then
        Add a_t to the list L_j if it is not already present.
    end
    If |L_j| > B for any j, delete the list L_j, and never add any items to it again.
end
Let i be the largest index such that |L_i| ≥ (1/5)B.
Return 2^{i-1}|L_i| as the estimate of ||f||_0
```

We initialize lists $L_0,L_1,\ldots,L_t\leftarrow\emptyset$, for $t=\Theta(\log n)$, and hash functions $H:[n]\to[2^\ell]$, where ℓ is set so that $n^2\le 2^\ell\le n^3$. The lists L_i will store a set of identities $L_i\subset[n]$ which have occurred in the stream. We also set $B\leftarrow\Theta(\frac{1}{\varepsilon^2}(\log\log n+\log\delta^{-1}))$. For now, assume that H is fully independent.

At each step when we see an update $a_i\in[n]$ (corresponding to an update which increments the value of f_i by one), we compute j such that $2^{\ell-j-1}\le H(a_i)\le 2^{\ell-j}$. Note that this event occurs with probability $2^{-(j+1)}$. Then we add the $O(\log n)$-bit identity a_i to the list L_j if $|L_j|<B$. Once $|L_k|=B$ for any $k\in[t]$, we delete the entire list L_k, and never add an item to L_k again. We call such a list L_k *saturated*. At the end of the stream, we find the largest value i such that $\frac{1}{5}B\le|L_i|$, and output $2^{i-1}|L_i|$ as our estimate of $\|f\|_0$.

We now analyze the above algorithm. Let i_0 be the smallest index such that $\mathbb{E}[|L_{i_0}|]\le\|f\|_0 2^{-(i_0+1)}<\frac{1}{5(1+\varepsilon)}B$. Note here that $\mathbb{E}[|L_k|]=2^{-(k+1)}\|f\|_0$ for any $k\in[t]$. By Chernoff bounds, with probability $1-\exp(-B)<1-\delta^2/\log(n)$ we have that $|L_{i_0}|<\frac{1}{5}B$. We can then union bound over all such indices $i\ge i_0$. This means that we will not output the estimate used from any index $i\ge i_0$. Similarly, by a Chernoff we have that $|L_{i_0-1}|=(1\pm\varepsilon)\|f\|_0 2^{-i_0}<\frac{2}{5}B$ and $|L_{i_0-2}|=(1\pm\varepsilon)\|f\|_0 2^{-i_0+1}$, and moreover we have $\frac{2}{5(1+\varepsilon)}B\le\|f\|_0 2^{-i_0+1}\le\frac{4}{5}B$, meaning that the output of our algorithm will be either $|L_{i_0-1}|2^{i_0-2}$ or $|L_{i_0-2}|2^{i_0-3}$, each of which yields a $(1\pm\varepsilon)$ estimate. Now note that we cannot store a fully independent hash function H, but since we only needed all events to hold with probability $1-\Theta(\delta^2/\log(n))$, it suffices to choose H to

[8] We remark that it is possible to optimize the $\log n$ factor to $O(\log\delta^{-1}+\log\varepsilon^{-1}+\log\log n$ by hashing the identities stored in the lists of the algorithm to a domain of size $\mathrm{poly}(\delta^{-1},\varepsilon^{-1},\log n)$. However, in our application we will be setting $\delta\ll 1/n$, and so the resulting adversarially robust algorithm would actually be *less* space efficient.

be a d-wise independent hash function for $d = O(\log\log n + \log\delta^{-1})$, which yields Chernoff-style tail inequalities with a decay rate of $\exp(-\Omega(d))$ (see e.g. Theorem 5 of [35]).

Now to analyze the space bound. Trivially, we store at most $O(\log n)$ lists L_i, each of which stores at most B identities which require $O(\log n)$ bits each to store, yielding a total complexity of $O(\frac{1}{\varepsilon^2}\log^2 n(\log\log n + \log\delta^{-1}))$. We now show however that at any given step, there are at most $O(B\log\log n)$ many identities stored in all of the active lists. To see this, let $i_0 < i_1 < \cdots < i_s$ be the time steps such that $\|f^{(i_j)}\|_0 = 2^{j+1}\cdot B$, and note that $s \le \log(n)+1$. Note that before time i_0, at most B identities are stored in the union of the lists. First, on time step i_j for any $j \in [s]$, the expected size of $|L_{j-2}|$ is at least $2|B|$ (had we never deleted saturated lists), and, with probability $1-(\delta/\log n)^{10}$ after a union bound, it holds that $|L_{j'}|$ is saturated for all $j' \le j-2$. Moreover, note that the expected number of identities written to lists $L_{j'}$ with $j' \ge j-1$ is $\|f^{(i_j)}\|_0\sum_{\nu\ge 1}2^{-j+1+\nu} \le 2B$, and is at most $4B$ with probability at least $1-(\delta/\log n)^{10}$ (using the d-wise independence of H). We conclude that on time step t_j, the total space being used is $O(B\log n)$ with probability at least $1-(\delta/\log n)^{10}$, so we can union bound over all such steps i_j for $j \in [s]$.

Next, we must analyze the space usage at steps τ for $i_j < \tau < i_{j+1}$. Note that the number of new distinct items which occur over all such time steps τ is at most $2^{j+1}\cdot B$ by definition. Since we already conditioned on the fat that $|L_{j'}|$ is saturated for all $j' \le j-2$, it follows that each new item is written into a list with probability at most 2^{-j}. Thus the expected number of items which are written into lists within times τ satisfying $i_j < \tau < i_{j+1}$ is $2^j \cdot B\cdot 2^{-j} = B$ in expectation, and at most $4B$ with probability $1-(\delta/\log n)^{10}$ (again using the d-wise Independence of H). Conditioned on this, the total space used in these steps is at most $O(B\log n) = O(\frac{1}{\varepsilon^2}\log n(\log\log n + \log\delta))$ in this interval, and we then can union bound over all such $O(\log n)$ intervals, which yields the desired space.

Update Time: Finally, for the update time, note that at each stream update $a_i \in [n]$ the first step of the algorithm. Naïvely, computing a d-wise independent hash function requires $O(d)$ arithmetic operations (in the standard RAM model), because H in this case is just a polynomial of degree d over $\mathbb{Z}$. On the other hand, we can batch sequences of $d = O(\log\log n + \log\delta^{-1})$ computations together, which require an additive $O(d\log n) = O(\log n(\log\log n + \log\delta^{-1}))$ bits of space at any given time step to store (which is dominated by the prior space complexity). Then by Proposition 5.3, all d hash function evaluations can be carried out in $O(d\log^2 d\log\log d) = O(d\log^2(\log\frac{\log n}{\delta})\log\log\log\log\frac{n}{\delta})$ time. The work can then be evenly distributed over the following d steps, giving a worst case update time of $O(\log^2(\log\frac{\log n}{\delta})\log\log\log\log\frac{n}{\delta})$. Note that this delays the reporting of the algorithm for the contribution of updates by a total of d steps, causing an additive d error. However, this is only an issue if $d \ge \varepsilon\|f\|_0$, which occurs only when $\|f\|_0 \ge \frac{1}{\varepsilon}d$. Thus for the first $D = O(\varepsilon^{-1}d)$ distinct items, we can store the non-zero items exactly (and deterministically), and use the output of this deterministic algorithm. The space required for this is $O(\varepsilon^{-1}\log(n)(\log\log n + \log\delta^{-1})$, which is dominated by the space usage of the algorithm overall. After D distinct items have been seen, we switch over to using the output of the randomized algorithm described here. Finally, the only other operation involves adding an identities to at most one list per update, which is $O(1)$ time, which completes the proof. ■

We can use the prior result of Lemma 5.2, along with our argument for union bounding over adversarial computation paths of Lemma 3.8 and the flip number bound of Corollary 3.5, which results in an adversarially robust streaming algorithm for distinct elements estimation with extremely fast update time.

Theorem 5.4. *There is a streaming algorithm which, with probability $1-n^{-(C/\varepsilon)\log n}$ for any constant $C \ge 1$, when run on an adversarial chosen insertion-only data stream, returns a $(1\pm\varepsilon)$ multiplicative estimate of the number of distinct elements at every step in the stream. The space required is $O(\frac{1}{\varepsilon^3}\log^3 n)$, and the worst case running time is $O\left(\left(\log^2\log\frac{\log n}{\varepsilon}\right)\cdot\left(\log\log\log\frac{\log n}{\varepsilon}\right)\right)$ per update.*

6 Heavy Hitters

In this section, we study the popular *heavy-hitters* problem in data streams. The heavy hitters problem tasks the algorithm with recovering the largest items in a data-set. Stated simply, the goal is to report a list S of items f_i that appear least τ times, meaning $f_i \geq \tau$, for a given threshold τ. Generally, τ is parameterized in terms of the L_p norm of the frequency vector f, so that $\tau = \varepsilon \|f\|_p$. For $p > 2$, this problem is known to take polynomial space [29]. Thus, the strongest such guarantee that can be given in sub-polynomial space is known as the L_2 guarantee:

Definition 6.1. A streaming algorithm is said to solve the (ε, δ)-heavy hitters problem with the L_2 guarantee if the algorithm, when run on a stream with frequency vector $f \in \mathbb{R}^n$, outputs a set $S \subset [n]$ such that with probability $1-\delta$ the following holds: for every $i \in [n]$ if $|f_i| \geq \varepsilon \|f\|_2$ then $i \in S$, and if $|f_i| \leq (\varepsilon/2)\|f\|_2$ then $i \notin S$.

We will also introduce the related task of ε-point queries.

Definition 6.2. A streaming algorithm is said to solve the (ε, δ) point query problem with the L_2 guarantee if with probability $1 - \delta$, at every time step $t \in [m]$, for each coordinate $i \in [n]$ it can output an estimate $\widehat{f}_i^t$ such that $|\widehat{f}_i^t - f_i^{(t)}| \leq \varepsilon \|f^{(t)}\|_2$. Equivalently, it outputs a vector $\widehat{f}^t \in \mathbb{R}^n$ such that $\|f^{(t)} - \widehat{f}^t\|_\infty \leq \varepsilon \|f^{(t)}\|_2$.[9]

Notice that for any algorithm that solves the (ε, δ)-point query problem, if it also has estimates $R^t = (1 \pm \varepsilon/10)\|f^{(t)}\|_2$ at each time step $t \in [m]$, then it immediately gives a solution to the (ε, δ) heavy hitters problem by just outputting all $i \in [n]$ with $\tilde{f}_i^t > (3/4)\varepsilon R^t$. Thus solving (ε, δ)-point queries, together with F_2 tracking, is a stronger property. In the following, we say that $\widehat{f}^t$ is ε-correct at time t if $\|f^{(t)} - \widehat{f}^t\|_\infty \leq \varepsilon \|f^{(t)}\|_2$.

In this section, we demonstrate how this fundamental task of point query estimation can be accomplished robustly in the adversarial setting. Note that we have already shown how F_2 tracking can be accomplished in the adversarial model, so our focus will be on point queries. Our algorithm relies on a similar sketch switching technique as used in Lemma 3.6, which systematically hides randomness from the adversary by only publishing a new estimate $\widehat{f}^t$ when absolutely necessary. To define what is meant by "absolutely necessary", we will first need the following Proposition.

Proposition 6.3. *Suppose that $\widehat{f}^t \in \mathbb{R}^n$ is ε-correct at time t on an insertion only stream, and let $t_1 > t$ be any time step such that $\|f^{(t_1)} - f^{(t)}\|_2 \leq \varepsilon \|f^{(t)}\|_\infty$. Then $\widehat{f}^t$ is 2ε-correct at time t_1.*

Proof. $\|\widehat{f}^t - f^{(t_1)}\|_\infty \leq \|\widehat{f}^t - f^{(t)}\|_\infty + \|f^{(t_1)} - f^{(t)}\|_\infty \leq \varepsilon \|f^{(t)}\|_2 + \varepsilon \|f^{(t)}\|_2 \leq 2\varepsilon \|f^{(t_1)}\|_2$. ■

To prove the main theorem of Section 6, we will need the classic *count-sketch* algorithm for finding L_2 heavy hitters [10], which solves the more general point query problem in the static setting with high probability.

Lemma 6.4 ([10])**.** *There is a streaming algorithm in the non-adversarial insertion only model which solves the (ε, δ) point query problem, using $O(\frac{1}{\varepsilon^2} \log n \log \frac{n}{\delta})$ bits of space.*

We are now ready to prove the main theorem of this section.

Theorem 6.5. *Fix any $\varepsilon, \delta > 0$. There is a streaming algorithm in the adversarial insertion only model which solves the (ε, n^{-C}) point query problem, and also the $O(\varepsilon, n^{-C})$-heavy hitters problem, for any constant $C > 1$. The algorithm uses $O(\frac{\log \varepsilon^{-1}}{\varepsilon^3} \log^2 n)$ bits of space.*

[9] We note that a stronger form of error is possible, called the *tail guarantee*, which does not count the contribution of the top $1/\varepsilon^2$ largest coordinates to the error $\varepsilon \|f\|_2$. We restrict to the simpler version of the L_2 guarantee.

Proof. Since we already know how to obtain estimates $R^t = (1 \pm \varepsilon/100)\|f^{(t)}\|_2$ at each time step $t \in [m]$ in the adversarial insertion only model within the required space, it will suffice to show that we can obtain estimates $\widehat{f^t}$ which are ε-correct at each time step t (i.e., it will suffice to solve the point query problem). Let $1 = t_1, t_2, \ldots, t_T = m$ for $T = \Theta(\varepsilon^{-1} \log n)$ be any set of time steps such that $\|f^{(a_{t+1})} - f^{(a_t)}\|_2 \leq \varepsilon \|f^{(a_t)}\|_2$ for each $i \in [T-1]$. Then by Proposition 6.3, we know that if we output a estimate $\widehat{f^i}$ on time t_i which is ε-correct for time t_i, then $\widehat{f^i}$ will still be 2ε correct at time t_{i+1}. Thus our approach will be to output vectors $\widehat{f^1}, \widehat{f^2}, \ldots, \widehat{f^T}$, such that we output the estimate $\widehat{f^i} \in \mathbb{R}^n$ at all times τ such that $t_i \leq \tau_i < t_{i+1}$, and such that $\widehat{f^i}$ is ε-correct for time t_i.

First, to find the time steps t_i, we run the adversarially robust F_2 estimator of Theorem 4.1, which gives an estimate $R^t = (1 \pm \varepsilon/100)\|f^{(t)}\|_2$ at each time step $t \in [m]$ with probability $1 - n^{-C}$ for any constant $C > 1$, and uses space $O(\varepsilon^{-3} \log^2 n \log \varepsilon^{-1})$. Notice that this also gives the required estimates R^t as stated above. By using an $\varepsilon/2$-rounding (Definition 3.7) of the output of this F_2 estimation algorithm, we obtain our desired points t_i. Notice that this also gives $T = \Theta(\varepsilon^{-1} \log n)$ as needed, by the flip number bound of Corollary 3.5. Next, to obtain the desired ε point query estimators at each time step t_i, we run T independent copies of the point query estimation algorithm of Lemma 6.4. At time t_i, we use the output vector of the i-th copy as our estimate $\widehat{f^i}$, which will also be used without any modification on all times τ with $t_i \leq \tau < t_{i+1}$. Since each copy of the algorithm only reveals any of its randomness at time t_i, at which point it is never used again, by the same argument as Lemma 3.6 it follows that each $\widehat{f^i}$ will be ε-correct for time t_i. Namely, since the set of stream updates on times $1, 2, \ldots, t_i$ are independent of the randomness used in the i-th copy of point-estimation algorithm, we can deterministically fix the updates on these time steps, and condition on the i-th copy of the non-adversarial streaming algorithm being correct on these updates. Therefore this algorithm correctly solves the 2ε point query problem on an adversarial stream. The total space used is

$$O\left(\varepsilon^{-3} \log^2 n \log \varepsilon^{-1} + T\varepsilon^{-2} \log^2 n\right)$$

We now note that we can improve the space by instead running only $T' = O(\varepsilon^{-1} \log \varepsilon^{-1})$ independent copies of the algorithm of Lemma 6.4. Each time we use one of the copies to output the desired estimate $\widehat{f^i}$, we completely restart that algorithm on the remaining suffix of the stream, and we loop modularly through all T' copies of the algorithm, at each step using the copy that was least recently restarted to output an estimate vector. More formally, we keep copies $\mathcal{A}_1, \ldots, \mathcal{A}_{T'}$ of the of the algorithm of Lemma 6.4. Each time we arrive at a new step t_i and must produce a new estimate $\widehat{f^i}$, we query the algorithm $\mathcal{A}_j$ that was *least recently restarted*, and use the estimate obtained by that algorithm, along with the estimates R^t.

The same correctness argument will hold as given above, except now each algorithm, when used after being restarted at least once, will only be ε-correct for the frequency vector defined by a *suffix* of the stream. However, by the same argument used in Theorem 4.1, we can safely disregard the prefix that was missed by this copy of the algorithm, because it contains only an $\varepsilon/100$-fraction of the total L_p mass of the current frequency vector when it is applied again. Formally, if an algorithm is used again at time t_i, and it was last restarted at time τ, then by the correctness of our estimates R^t, the L_2 norm must have gone up by a factor of $(1+\varepsilon)^{T'} = \frac{100}{\varepsilon}$, so $\|f^{(\tau)}\|_2 \leq \varepsilon/100 \|f^{(t_i)}\|_2$. Moreover, we have that the estimate $\widehat{f^i}$ produced by this copy satisfies $\|\widehat{f^i} - (f^{(t_i)} - f^{(\tau)})\|_\infty \leq \varepsilon \|f^{(t_i)} - f^{(\tau)}\|_2$. But then

$$\begin{aligned}
\|\widehat{f^i} - f^{(t_i)}\|_\infty &\leq \|\widehat{f^i} - (f^{(t_i)} - f^{(\tau)})\|_\infty + \|f^{(\tau)}\|_\infty \\
&\leq \varepsilon \|f^{(t_i)} - f^{(\tau)}\|_2 + \|f^{(\tau)}\|_2 \\
&\leq \varepsilon \left(\|f^{(t_i)}\|_2 + \|f^{(\tau)}\|_2\right) + \varepsilon/100 \|f^{(t_i)}\|_2 \qquad (1) \\
&\leq \varepsilon \|f^{(t_i)}\|_2 (1 + \varepsilon) + \varepsilon/100 \|f^{(t_i)}\|_2 \\
&\leq 2\varepsilon \|f^{(t_i)}\|_2
\end{aligned}$$

Thus $\widehat{f^i}$ is still 2ε-correct at time t_i for the full stream vector $f^{(t_i)}$. So by the same argument as above using Proposition 6.3, it follows that the output of the overall algorithm is always 4ε-correct for all time steps $\tau \in [m]$, and we can then resale ε by a factor of $1/4$. Substituting the new number T' of copies used into the above equation, we obtain the desired complexity. ■

7 Entropy Estimation

We now show how our general techniques developed in Section 3 can be used to approximate the empirical Shannon entropy $H(f)$ of an adversarial stream. Recall that for a non-zero vector f, we have that $H(f) = -\sum_{i,f_i \neq 0} p_i \log(p_i)$, where $p_i = \frac{|f_i|}{\|f\|_1}$. Also recall that for $\alpha > 0$, the α-Renyi Entropy $H_\alpha(x)$ of x is given by $H_\alpha(x) = \log\left(\frac{\|x\|_\alpha^\alpha}{\|x\|_1^\alpha}\right)/(1-\alpha)$. The proofs omitted from this section can be found in Appendix **??**.

We begin with the following observation, which will allow us to consider multiplicative approximation of $2^{H(x)}$. Then, by carefully bounding the flip number of the Renyi entropy H_α for α close to 1, we will be able to bound the flip number of H.

Remark. Note that any algorithm that gives an ε-additive approximation of the Shannon Entropy $H(x) : \mathbb{R}^n \to \mathbb{R}$ gives a $(1 \pm \varepsilon)$ multiplicative approximation of $g(x) = 2^{H(x)}$, and vice-versa.

Proposition 7.1 (Theorem 3.1 of [21])**.** *Let $x \in \mathbb{R}^n$ be a probability distribution whose smallest non-zero value is at least $\frac{1}{m}$, where $m \geq n$. Let $0 < \varepsilon < 1$ be arbitrary. Define $\mu = \varepsilon/(4\log m)$ and $\nu = \varepsilon/(4\log n \log m)$, $\alpha = 1 + \mu/(16\log(1/\mu))$ and $\beta = 1 + \nu/(16\log(1/\nu))$. Then*

$$1 \leq \frac{H_\alpha}{H} \leq 1 + \varepsilon \text{ and } 0 \leq H - H_\beta \leq \varepsilon$$

Proposition 7.2. *Let $g : \mathbb{R}^N \to R$ be $g(x) = 2^{H(x)}$, i.e., the exponential of the Shannon entropy. Then the (ε, m)-flip number of g for the insertion only streaming model is $\lambda_{\varepsilon,m}(g) = O(\frac{1}{\varepsilon^3}\log^3 m)$.*

The proof of the above proposition is given later in this section. We now state the main result on adversarially robust entropy estimation.

Theorem 7.3 (Robust Additive Entropy Estimation)**.** *There is an algorithm for ε-additive approximation of entropy in the insertion-only adversarial streaming model using $O(\frac{1}{\varepsilon^5}\log^4 n)$-bits of space in the random oracle model, and $O(\frac{1}{\varepsilon^5}\log^6 n)$-bits of space in the general insertion-only model.*

To obtain our entropy estimation algorithm of Theorem 7.3, we will first need to state the results for the state of the art non-adversarial streaming algorithms for additive entropy estimation. The first algorithm is a $O(\varepsilon^{-2}\log^2 n)$-bit streaming algorithm for additive approximation of the entropy of a turnstile stream, which in particular holds for insertion only streams. The second result is a $\tilde{O}(1/\varepsilon^2)$ upper bound for entropy estimation in the insertion only model when a random oracle is given. Recall that the random oracle model of streaming means that the algorithm is given random (read-only) access to an arbitrarily large string of random bits.

Lemma 7.4 ([11])**.** *There is an algorithm in the strict turnstile model that gives a ε-additive approximation to the Shannon Entropy $H(f)$ of the stream. The failure probability is δ, and the space required is $O(\frac{1}{\varepsilon^2}\log^2 n \log \delta^{-1})$ bits.*

Lemma 7.5 ([23])**.** *There is an algorithm in the insertion-only random oracle model that gives a ε-additive approximation to the Shannon Entropy $H(f)$ of the stream. The failure probability is δ, and the space required is $O(\frac{1}{\varepsilon^2}(\log \delta^{-1} + \log\log n + \log \varepsilon^{-1}))$*

We now give the proof of Proposition 7.2, and then the main Theorem 7.3.

Proof of Proposition 7.2. By Proposition 7.1, it suffices to get a bound on the flip number of H_β for the parameters $\beta = 1 + \nu/(16\log(1/\nu))$ and $\nu = \varepsilon/(4\log n \log m)$. Recall $g(x) = 2^{H_\beta(x)} = (\|x\|_\beta^\beta/\|x\|_1^\beta)^{1/(1-\beta)}$. Now to increase $g(x)$ by a factor of $(1+\varepsilon)$, one must increase $\|x\|_\beta^\beta/\|x\|_1^\beta$ by a factor of $(1+\Theta(\varepsilon(1-\beta))$. For this to happen, $\|x\|_\beta$ must increase by a factor of $(1+\Theta(\varepsilon(1-\beta))$. Since $\|x\|_\beta \le \|x\|_1 \le n^{1-1/\beta}\|x\|_1 = (1+O(\varepsilon/\log n))\|x\|_1$, and thus $\|x\|_1$ must also increase by a factor of $(1+\Theta(\varepsilon)(1-\beta))$.

Similarly, for $g(x)$ to decrease by a factor of $(1+\varepsilon)$, this requires $\|x\|_1$ to increase by a factor of $(1+\Theta(\varepsilon(1-\beta))$. In summary, for $g(x)$ to change by a factor of $(1\pm\varepsilon)$, $\|x\|_1$ must increase by a factor of $(1+\Theta(\varepsilon(1-\beta)) = (1+\tau)$, where $\tau = \tilde{\Theta}(\varepsilon^2/\log^2 n)$. $\|f^{(m)}\|_1 \le Mn$ and $\|\cdot\|_1$ is monotone for insertion only streams, it follows that this can occur at most $O(\frac{\log^3 n}{\varepsilon^2})$ times, which completes the proof since $\log n = \Theta(\log m)$. ■

Proof of Theorem 7.3. The proof follows directly from an application of Lemma 3.6, using the non-adversarial algorithms of Lemmas 7.4 and 7.5, as well as the flip number bound of Lemma 7.2. Note that to turn the algorithms of Lemmas 7.4 and 7.5 into tracking algorithms, one must set $\delta < 1/m$, which yields the stated complexity. ■

8 Bounded Deletion Streams

In this section, we show how our results can be used to obtain adversarially robust streaming algorithms for the *bounded-deletion model*, introduced in [22]. The bounded deletion model serves as an intermediate model between the turnstile and insertion only model. Motivated by common lower bounds for turnstile streams, which utilize seemingly unrealistic hard instances that insert a large number of items before deleting nearly all of them, bounded deletion streams are possibly a more representative model for real world data-streams. Intuitively, a bounded deletion stream is one where the F_p moment of the stream is a $\frac{1}{\alpha}$ fraction of what the F_p moment would have been had all updates been replaced with their absolute values, meaning that the stream does not delete off an arbitrary amount of the F_p weight that it adds over the course of the stream. Formally, the model is as follows.

Definition 8.1. Fix any $p \ge 1$ and $\alpha \ge 1$. A data stream $u_1, \dots, u_m$, where $u_i = (a_i, \Delta_i) \in [n] \times \{1, -1\}$ are the updates to the frequency vector f, is said to be an F_p α-bounded deletion stream if at every time step $t \in [m]$ we have $\|f^{(t)}\|_p^p \ge \frac{1}{\alpha}\sum_{i=1}^n (\sum_{t'<t:a_{t'}=i} |\Delta_{t'}|)^p$.

Specifically, the α-bounded deletion property says that the F_p moment $\|f^{(t)}\|_p$ of the stream is at least $\frac{1}{\alpha}\|h^{(t)}\|_p$, where h is the frequency vector of the stream with updates $u_i' = (a_i, \Delta_i')$ where $\Delta_i' = |\Delta_i|$ (i.e., the absolute value stream). Note here that the model assumes unit updates, i.e., we have $|\Delta_i| = 1$ for each $i \in [m]$, which can be accomplished without loss of generality with respect to the space complexity of algorithms, by simply duplicating integral updates into unit updates.

In [22], the authors show that for α-bounded deletion streams, a factor of $\log n$ in the space complexity of turnstile algorithms can be replaced with a factor of $\log \alpha$ for many important streaming problems. In this section, we show another useful property of bounded-deletion streams: norms in such streams have bounded flip number. We use this fact to design adversarially robust streaming algorithms for data streams with bounded deletions.

Lemma 8.2. *Fix any $p \ge 1$. The $\lambda_{\varepsilon,m}(\|\cdot\|_p)$ flip number of the L_p norm of a α-bounded deletion stream is at most $O(p\frac{\alpha}{\varepsilon^p}\log n)$*

Proof. Let h be the frequency vector of the stream with updates $u_i' = (a_i, \Delta_i')$ where $\Delta_i' = |\Delta_i|$. Note that h is then the frequency vector of an insertion only stream. Now let $0 \le t_1 < t_2 <$

$\cdots < t_k \leq m$ be any set of time steps such that $\|f^{(t_i)}\|_p \notin (1 \pm \varepsilon)\|f^{(t_{i+1})}\|_p$ for each $i \in [k-1]$. Since by definition of the α-bounded deletion property, we have $\|f^{(t)}\|_p \geq \frac{1}{\alpha^{1/p}}\|h^{(t)}\|_p$ for each $t \geq T$, it follows that

$$\|f^{(t_{i+1})} - f^{(t_i)}\|_p \geq \left|\|f^{(t_{i+1})}\|_p - \|f^{(t_i)}\|_p\right| \geq \varepsilon\|f^{(t_{i+1})}\|_p \geq \frac{\varepsilon}{\alpha^{1/p}}\|h^{(t_{i+1})}\|_p \geq \frac{\varepsilon^p}{\alpha^{1/p}}\|h^{(t_i)}\|_p \quad (2)$$

Where in the last inequality we used the fact that h is an insertion only stream. Now since the updates to h are the absolute value of the updates to f, we also have that $\|h^{(t_{i+1})} - h^{(t_i)}\|_p^p \geq \|f^{(t_{i+1})} - f^{(t_i)}\|_p^p \geq \frac{\varepsilon^p}{\alpha}\|h^{(t_i)}\|_p^p$. Thus

$$\|h^{(t_{i+1})}\|_p^p = \|h^{(t_i)} + \left(h^{(t_{i+1})} - h^{(t_i)}\right)\|_p^p \geq \|h^{(t_i)}\|_p^p + \|h^{(t_{i+1})} - h^{(t_i)}\|_p^p \geq (1 + \frac{\varepsilon^p}{\alpha})\|h^{(t_i)}\|_p^p \quad (3)$$

Where in the second inequality, we used the fact that $\|X+Y\|_p^p \geq \|X\|p^p + \|Y\|_p^p$ for non-negative integral vectors X, Y when $p \geq 1$. Thus $\|h^{(t_{i+1})}\|_p^p$ must increase by a factor of $(1+\varepsilon^p/\alpha)$ from $\|h^{(t_i)}\|_p^p$ whenever $\|f^{(t_i)}\|_p \notin (1 \pm \varepsilon)\|f^{(t_{i+1})}\|_p$. Since $\|0\|_p^P = 0$, and $\|h^{(m)}\|_p^p \leq M^p n \leq n^{cP}$ for some constant $c > 0$, it follows that this can occur at most $O(p\frac{\alpha}{\varepsilon^p}\log n)$ many times. Thus $k = O(p\frac{\alpha}{\varepsilon^p}\log n)$, which completes the proof. ■

We now use our computation paths technique of Lemma 3.8, along with the space optimal turnstile F_p estimation algorithm of [27], to obtain adversarially robust algorithms for α-bounded deletion streams. Specifically, we show that we can estimate the F_p moment of a bounded deletion stream robustly. We remark that once F_2 moment estimation can be done, one can similarly solve the heavy hitters problem in the robust model using a similar argument as in Section 6, except without the optimization used within the proof of Theorem 6.5 which restarts sketches on a suffix of the stream. The result the space would be precisely a $\frac{\alpha}{\varepsilon}\log n$ factor larger than the space stated in Theorem 6.5.

Theorem 8.3. *Fix $p \in [1,2]$ and any constant $C > 1$. Then there is an adversarially robust F_p estimation algorithm which, with probability $1 - n^{-C}$, returns at each time step $t \in [m]$ an estimate R^t such that $R^t = (1 \pm \varepsilon)\|f^{(t)}\|_p^p$. The space used by the algorithm is $O(\alpha\varepsilon^{-(2+p)}\log^3 n)$.*

Proof of Theorem 8.3. We use the turnstile algorithm of [27], which gives a estimate $R^t = (1 \pm \varepsilon)\|f^{(t)}\|_p^p$ at a single point $t \in [m]$ with probability $1-\delta$, using $O(\varepsilon^{-2}\log n \log \delta^{-1})$ bits of space. We can set $\delta = 1/\text{poly}(m)$, and union bound over all steps, to obtain that $R^t = (1 \pm \varepsilon)\|f^{(t)}\|_p^p$ at all time steps $t \in [m]$ with probability $1 - n^{-C}$. Thus this gives a strong F_p tracking algorithm using $O(\varepsilon^{-2}\log n \log(n/\delta))$ bits of space. The theorem then follows from applying Lemma 3.8, along with the flip number bound of Lemma 8.2. ■

9 Adversarial Attack Against the AMS Sketch

It was shown by Hardt and Woodruff [20] that linear sketches can in some cases be vulnerable to adaptive adversarial attacks (see Subsection 1.2). In this section, we show another instance of this phenomenon, demonstrating that the classic Alon-Matias-Szegedy (AMS) sketch [3] for estimating the L_2 norm of a data stream is inherently non-robust. To this end, we describe an attack fooling the AMS sketch into outputting a value which is not a good approximation of the norm $\|f\|_2^2$ of the frequency vector. Our attack provides an even stronger guarantee: for any $r \geq 1$ and an AMS sketch with r/ε^2 rows, our adversary needs to only create $O(r)$ adaptive stream updates before it can fool the AMS sketch into outputting an incorrect result.

We first recall the AMS sketch for estimating the L_2 norm. The AMS sketch generates (implicitly) a random matrix $A \in \mathbb{R}^{t \times n}$ such that the entries $A_{i,j} \sim \{-1, 1\}$ are i.i.d. Rademacher.[10]

[10] In fact, the AMS sketch works even if the entries within a row of A are only 4-wise independent. Here, we show an attack against the AMS sketch if it is allowed to store a fully independent sketch A.

The algorithm stores the sketch $Af^{(j)}$ at each time step j, and since the sketch is linear it can be updated throughout the stream: $Af^{(j+1)} = Af^{(j)} + A \cdot e_{i_{j+1}} \Delta_{j+1}$ where (i_{j+1}, Δ_{j+1}) is the $j+1$-st update. The estimate of the sketch at time j is $\frac{1}{t}\|Af^{(j)}\|_2^2$, which is guaranteed to be with good probability a $(1 \pm \varepsilon)$ estimate of $\|f^{(j)}\|_2^2$ in non-adversarial streams if $t = \Theta(\varepsilon^{-2})$.

We now describe our attack. Let S be a $t \times n$ Alon-Matias-Szegedy sketch. Equivalently, $S_{i,j}$ is i.i.d. uniformly distributed in $\{-t^{-1/2}, t^{-1/2}\}$, and the estimate of AMS is $\|Sf^{(j)}\|_2^2$ at the j-th step. The protocol for the adversary is as follows. In the following, we let $e_i \in \mathbb{R}^n$ denote the standard basis vector which is zero everywhere except the i-th coordinate, where it has the value 1.

Algorithm 3: Adversary for AMS sketch

```
1  w ← C · √t · e_1
2  for i = 2, ..., m do
3  |   old ← ||Sw||_2^2
4  |   w ← w + e_i
5  |   new ← ||Sw||_2^2
6  |   if new − old < 1 then
7  |   |   w ← w + Se_i
8  |   end
9  |   else if new − old = 1 then
10 |   |   with probability 1/2, set w ← w + Se_i
11 |   end
12 end
```

Note that the vector w in the above algorithm is always equal to the current frequency vector of the stream, namely $w = f^{(j)}$ after the j-th update. Note that the above algorithm can be implemented by an adversary who only is giving the estimate $\|Sw\|_2^2 = \|Sf^{(j)}\|_2^2$ of the AMS sketch after every step j in the stream. To see this, note that the algorithm begins by inserting the first item $(i_1, \Delta_1) = (1, C \cdot \sqrt{t})$ for a sufficiently large constant C. Next, for $i = 2, \ldots, n$, it inserts the item $i \in [n]$ once if doing so increases the estimate of AMS by more than 1. If the estimate of AMS is increased by less than 1, it inserts the item i twice (i.e., it inserts an update $(i, 2) \in [n] \times \mathbb{Z}$). Lastly, if inserting the item $i \in [n]$ increases the estimate of AMS by *exactly* 1, the adversary chooses to insert $i \in [n]$ once with probability $1/2$, otherwise it inserts $i \in [n]$ twice.

We now claim that at the end of a stream of $m = O(t)$ updates, with good probability $\|w\|_2^2 = \|Sf^{(m)}\|_2^2 \notin (1 \pm \varepsilon)\|f^{(m)}\|_2^2$. In fact, we show that regardless of the number of rows t in the AMS sketch, we force the AMS to give a solution that is not even a 2-approximation.

Theorem 9.1. *Let $S \in \mathbb{R}^{t \times n}$ be an AMS sketch (i.i.d. Rademacher matrix scaled by $t^{-1/2}$), where $1 \le t < n/c$ for some constant c. Suppose further that the adversary performs the adaptive updates as described in Algorithm 3. Then with probability $9/10$, by the m-th stream update for some $m = O(t)$, the AMS estimate $\|Sf^{(m)}\|_2^2$ of the norm $\|f^{(m)}\|_2^2$ of the frequency vector f defined by the stream fails to be a $(1 \pm 1/2)$ approximation of the true norm $\|f^{(m)}\|_2^2$. Specifically, we will have $\|Sf^{(m)}\|_2^2 < \frac{1}{2}\|f^{(m)}\|_2^2$.*

Proof. For $j = 2, 3 \ldots$ we say that the j-th *step* of Algorithm 3 is the step in the for loop where the parameter i is equal to j, and we define the first step to just be the state of the stream after line 1 of Algorithm 3. Let w^i be the state of the frequency vector at the end of the i-th step of the for loop in Algorithm 3, let $y^i = Sw^i$ be the AMS sketch at this step, and let $s_i = \|Sw^i\|_2^2$ be the estimate of AMS at the same point. Note that we have $w^1 = C \cdot \sqrt{t} \cdot e_1$ for a sufficiently large constant C, and thus $s_1 = C^2 t$. That is, already on the first step of the

algorithm we have $\|w^1\|_2^2 = C^2 t$, and moreover since the stream is insertion only, we always have $\|w^i\|_2^2 \geq C^2 t$. Thus, it suffices to show that with good probability, at some time step $i \geq 2$ we will have $s_i < C^2 t/2$.

First, note that at any step $i = 2, 3, \ldots$, if we add Se_{i+1} once, we have $s_{i+1} = \|y^i + Se_{i+1}\|_2^2 = \sum_{j=1}^t ((y_j^i)^2 + 2y_j^i S_{j,i+1} + 1/t) = s_i + 1 + 2\sum_{j=1}^t y_j^i S_{j,i+1}$. If we add Se_{i+1} twice, we have $s_{i+1} = \|y^i + 2Se_{i+1}\|_2^2 = s_i + 4 + 4\sum_{j=1}^t y_j^i S_{j,i+1}$. By definition of the algorithm, we choose to insert Se_{i+1} twice if $\|y^i + Se_{i+1}\|_2^2 - s_i = 1 + 2\sum_{j=1}^t y_j^i S_{j,i+1} < 1$, or more compactly whenever $\sum_{j=1}^t y_j^i S_{j,i+1} < 0$. If $\sum_{j=1}^t y_j^i S_{j,i+1} > 0$, we insert Se_{i+1} only once. Finally, if $\sum_{j=1}^t y_j^i S_{j,i+1} = 0$, we flip an unbiased coin, and choose to insert Se_{i+1} either once or twice with equal probability 1/2. Now observe that the random variable $\sum_{j=1}^t y_j^i S_{j,i+1}$ is symmetric, since for any fixed y^i the $S_{j,i+1}$'s are symmetric and independent. Thus, we have that

$$\begin{aligned} \mathbb{E}\Big[\Big|\sum_{j=1}^t y_j^i S_{j,i+1}\Big|\Big] &= \mathbb{E}\Big[\sum_{j=1}^t y_j^i S_{j,i+1} \mid Se_{i+1} \text{ inserted once}\Big] \\ &= -\mathbb{E}\Big[\sum_{j=1}^t y_j^i S_{j,i+1} \mid Se_{i+1} \text{ inserted twice}\Big] \end{aligned} \tag{4}$$

Now recall that the vector $S_{*,i+1}$ given by the $(i+1)$-st column of S is just an i.i.d. Rademacher vector scaled by $1/\sqrt{t}$. Thus by the classic Khintchine inequality [19], we have that $\mathbb{E}[|\sum_{j=1}^t y_j^i S_{j,i+1}|] = \frac{1}{\sqrt{t}} \cdot \alpha \cdot \|y^i\|_2 = \alpha\sqrt{s_i}/\sqrt{t}$ for some absolute constant $\alpha > 0$ (in fact, $\alpha \geq 1/\sqrt{2}$ suffices by Theorem 1.1 of [19]). Putting these pieces together, the expectation of the estimate of AMS is then as follows:

$$\begin{aligned} \mathbb{E}[s_{i+1}] &= \frac{1}{2}(s_i + 1 + 2\alpha\frac{\sqrt{s_i}}{\sqrt{t}}) + \frac{1}{2}(s_i + 4 - 4\alpha\frac{\sqrt{s_i}}{\sqrt{t}}) \\ &= s_i + 5/2 - \alpha\sqrt{s_i/t} \\ &\leq s_i + 5/2 - \sqrt{s_i/2t} \end{aligned} \tag{5}$$

Where again the last line holds using the fact that $\alpha \geq 1/\sqrt{2}$. Thus $\mathbb{E}[s_{i+1}] = \mathbb{E}[s_i] + 5/2 - \mathbb{E}[\sqrt{s_i/2t}]$. First, suppose there exists some $i \leq C^2 t + 2$ such that $\mathbb{E}[\sqrt{s_i}] < C\sqrt{t/200}$. This implies by definition that $\sum_j \sqrt{j} \cdot \mathbf{Pr}[s_i = j] < C\sqrt{t/200}$, thus

$$\sqrt{C^2 t/2} \cdot \mathbf{Pr}[s_i \geq C^2 t/2] \leq \sum_{j \geq C^2 t/2} \sqrt{j} \cdot \mathbf{Pr}[s_i = j] < \sqrt{C^2 t/200} \tag{6}$$

Which implies that $\mathbf{Pr}[s_i \geq C^2 t/2] \leq 1/10$. Thus, on that step i, we have $\mathbf{Pr}[s_i < C^2 t/2] > 9/10$, and thus by time step i we have fooled the AMS sketch with probability at least 9/10. Thus, we can assume that for all $i = 2, 3, \ldots, (C^2 t + 2)$ we have $\mathbb{E}[\sqrt{s_i}] \geq C\sqrt{t/200}$. Setting $C > 200$, we have that $\mathbb{E}[s_{i+1}] < \mathbb{E}[s_i] - 1$ for all steps $i = 2, 3, \ldots, (C^2 t + 2)$ However, since $s_1 = C^2 t$, this implies that $\mathbb{E}[s_{C^2 t+2}] < -1$, which is impossible since s_j is always the value of a norm. This is a contradiction, which implies that such an i with $i \leq C^2 t + 2$ and $\mathbf{Pr}[s_i \geq C^2 t/2] \leq 1/10$ must exist, demonstrating that we fool the AMS sketch by this step with probability 9/10, which completes the proof. ■

10 Optimal Distinct Elements via Cryptographic Assumptions

Estimating the number of distinct elements (F_0-estimation) in a data stream is a fundamental problem in databases, network traffic monitoring, query optimization, data mining, and more. After a long line of work, [37, 28] settled space (and time) complexities of F_0-estimation by

giving an algorithm using $O(\varepsilon^{-2} + \log n)$ bits of space (with constant worst-case update time). The tracking version of this algorithm (where it outputs a correct estimate at each time step) takes memory $O(\varepsilon^{-2}(\log \varepsilon^{-1} + \log\log n) + \log n)$ bits and is also optimal [6].

However, these results only hold in the (standard) static setting. We show that using cryptographic tools (pseudorandom functions), we can transform this algorithm, using the same amount of memory to be robust in the adversarial setting as well, where the adversary is assumed to be *computationally bounded* (as opposed to our other results which have no assumptions on the adversary whatsoever).

The transformation actually works for a large class of streaming algorithms. Namely, any algorithm such that when given an element that appeared before, does not change its state at all (with probability 1). Since the F_0 tracking algorithm of [6] has this property, we can black-box apply our results to this algorithm.

First, we show how this transformation works assuming the existence of a truly random function, where the streaming algorithm has access to the function without needing to store it explicitly (the memory is free). Recall that this is known as the random oracle model of streaming, as such a function can be represented by a sufficiently long string of random bits. The model is appealing since we have different heuristic functions (such as the SHA-X family) that behave, as far as we can tell in practice, like random functions. Moreover, there is no memory cost when using them. Nevertheless, we discuss how to implement such a function with cryptographic tools (pseudorandom functions) and storing only a small secret key in the memory.

Theorem 10.1 (Robust Distinct Elements by Cryptographic Assumptions)**.** *In the random oracle model, there is an F_0-estimation (tracking) streaming algorithm in the adversarial setting, that for an approximation parameter ε uses $O(\varepsilon^{-2}(\log 1/\varepsilon + \log\log n) + \log n)$ bits of memory, and succeeds with probability $3/4$.*

Moreover, under a suitable cryptographic assumption, assuming the adversary has bounded running time of n^c, where m is the stream length and c is fixed, the random oracle can be replaced with a concrete function and the total memory is $O(\varepsilon^{-2}(\log 1/\varepsilon + \log\log n) + c\log n)$.

Proof. For simplicity, in the following proof we assume that we have a random permutation. We note that the proof with a random function is exactly the same conditioned on not having any collisions. If the random function maps the universe to a large enough domain (say of size at least m^2) then there will be no collisions with high probability. Thus, it suffices to consider permutations.

The solution is inspired by the work of [34] (which had a similar adaptive issue in the context of Bloom filters). Let Π be a truly random permutation, and let S be a tracking steaming algorithm with parameter ε. Let $L(\varepsilon, n)$ be the memory consumption of the algorithm. We construct an algorithm S' that works in the adversarial setting as follows. Upon receiving an element x the algorithm S' computes $x' = \Pi(x)$ and feeds it to S. The output of S' is exactly the output of S. Notice that applying Π to the stream does not change the number of distinct elements.

We sketch the proof. Assume towards a contradiction that there is adaptive adversary A' for S'. Consider the adversary A' at some point in time t, where the stream is currently $x_1, \ldots, x_t$. It has two options: (i) it can choose an element x_i, where $i \in [t]$ that appeared before, or (ii) it could choose a new element $x^* \notin \{x_1, \ldots, x_i\}$. Since the state of S' does not change when receiving duplicate items, and also does not change the number of distinct elements, option (i) has no effect on the success probability of A'. Thus, in order to gain a chance of winning A' must submit a new query. Thus, we can assume without loss of generality that A' submits only distinct elements.

For such an adversary A' let D_t be the distribution over states of S' at time t. Let D'_t be the distribution over states of S' for the fixed sequence $1, 2, \ldots, t$. We claim that $D_t \equiv D'_t$ (identical

distributions) for every $t \in [m]$. We show this by induction. The first query is non-adaptive, denote it by x_1. Then, since Π is a random permutation, we get that $\Pi(1) \equiv \Pi(x_1)$ which is what is fed to S. Thus, the two distribution are identical. Assume it holds for $t-1$. Consider the next query of the adversary (recall that we assumed that this is a new query). Then, for any x_t (that has not been previously queried by Π) the distribution of $\Pi(x_t) \equiv \Pi(t)$, and therefore we get that $D_t \equiv D'_t$.

Given the claim above, we get that A' is equivalent to a static adversary A that outputs $1, 2, \ldots, k$ for some $k \in [m]$. However, the choice of k might be adaptive. We need to show that S' works for all k simultaneously. Here we use the fact that S was a tracking algorithm (and thus also S'), which means that S' succeeds on every timestep. Thus, for the stream $1, 2, \ldots, m$ the algorithm S' succeeds at timestamp k which consists of k distinct elements. Thus, if there exists an adaptive choice of k that would make S' fail, then there would exist a point in time, k, such that S' fails at $1, \ldots, k$. Since S is tracking, such a point does not exist (w.h.p.).

For the second part of the theorem, we note that we can implement the random function using an exponentially secure pseudorandom function (see [17] for the precise definition and discussion). For a key K of size λ, the pesudorandom function $F_K(\cdot)$ looks random to an adversary that has oracle access to $F_K(\cdot)$ and runs in time at most $2^{\gamma\lambda}$ for some constant $\gamma > 0$. Let A be an adversary that runs in time at most n^c. Then, we set $O(\lambda = 1/\gamma \cdot c \cdot \log n)$ and get that A cannot distinguish between $F_K(\cdot)$ and the truly random function except when a negligible probability event occurs (i.e., the affect on δ is negligible and hidden in constants). Indeed, if A would be able to succeed against S' when using the oracle $F_K(\cdot)$, but, as we saw, it does not succeed when using a truly random function, then A' could be used to break the security of the pseudorandom function.

There are many different ways to concretely implement such a pseudorandom function with exponential security. First, one could use heuristic (and extremely fast) functions such as AES or SHA256 (see also [34] for a discussion on fast implementations of AES in the context of hash functions). Next, one can assume that the discrete logarithm problem (see [30] for the precise definition) over a group of size q is exponentially hard. Indeed, the best known algorithm for the problem runs in time $O(\sqrt{q})$. Setting $q \geq 2^\lambda$ gets us the desired property for $\gamma = 1/2$.

To complete the proof, we note that the only property of A we needed was that when given an element in the stream that has appeared before, A does not change its state at all. This property holds for many F_0 estimation algorithms, such as the one-shot F_0 algorithm of [28], and the F_0 tracking algorithm of [6]. Thus we can simply use the F_0 tracking algorithm of [6], which results in the space complexity as stated in the theorem. ■

Acknowledgments

The authors wish to thank Arnold Filtser for invaluable feedback. This work was done in part in the Simons Institute for the Theory of Computing.

References

[1] Ahn, K. J., Guha, S., and McGregor, A. Analyzing graph structure via linear measurements. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms* (2012), SIAM, pp. 459–467.

[2] Ahn, K. J., Guha, S., and McGregor, A. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems* (2012), ACM, pp. 5–14.

[3] Alon, N., Matias, Y., and Szegedy, M. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences 58*, 1 (1999), 137 – 147.

[4] Bar-Yossef, Z., Jayram, T. S., Kumar, R., and Sivakumar, D. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences 68*, 4 (2004), 702–732.

[5] Ben-Eliezer, O., and Yogev, E. The adversarial robustness of sampling. *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, to appear* (2020).

[6] Błasiok, J. Optimal streaming and tracking distinct elements with high probability. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms* (2018), SIAM, pp. 2432–2448.

[7] Błasiok, J., Ding, J., and Nelson, J. Continuous monitoring of l_p norms in data streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques* (2017), APPROX/RANDOM, pp. 32:1–32:13.

[8] Braverman, V., Chestnut, S. R., Ivkin, N., Nelson, J., Wang, Z., and Woodruff, D. P. Bptree: An ℓ_2 heavy hitters algorithm using constant memory. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (2017), ACM, pp. 361–376.

[9] Chakrabarti, A., and Kale, S. Strong fooling sets for multi-player communication with applications to deterministic estimation of stream statistics. In *IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)* (2016), pp. 41–50.

[10] Charikar, M., Chen, K., and Farach-Colton, M. Finding frequent items in data streams. *Theoretical Computer Science 312*, 1 (2004), 3–15.

[11] Clifford, P., and Cosma, I. A simple sketching algorithm for entropy estimation over streaming data. In *Artificial Intelligence and Statistics* (2013), pp. 196–206.

[12] DeWitt, D. J., Naughton, J. F., Schneider, D. A., and Seshadri, S. Practical skew handling in parallel joins. In *Proceedings of the 18th International Conference on Very Large Data Bases* (1992), VLDB, pp. 27–40.

[13] Ganguly, S. Deterministically estimating data stream frequencies. In *International Conference on Combinatorial Optimization and Applications* (2009), COCOA, Springer, pp. 301–312.

[14] Ganguly, S., and Woodruff, D. P. High probability frequency moment sketches. In *45th International Colloquium on Automata, Languages, and Programming* (2018), ICALP, pp. 58:1–58:15.

[15] Gilbert, A. C., Hemenway, B., Rudra, A., Strauss, M. J., and Wootters, M. Recovering simple signals. In *2012 Information Theory and Applications Workshop* (2012), IEEE, pp. 382–391.

[16] Gilbert, A. C., Hemenway, B., Strauss, M. J., Woodruff, D. P., and Wootters, M. Reusable low-error compressive sampling schemes through privacy. In *2012 IEEE Statistical Signal Processing Workshop (SSP)* (2012), IEEE, pp. 536–539.

[17] Goldreich, O. Foundations of cryptography - A primer. *Foundations and Trends in Theoretical Computer Science 1*, 1 (2005).

[18] Good, I. J. C332. surprise indexes and p-values. *Journal of Statistical Computation and Simulation 32*, 1–2 (1989), 90–92.

[19] Haagerup, U. The best constants in the khintchine inequality. *Studia Mathematica 70* (1981), 231–283.

[20] Hardt, M., and Woodruff, D. P. How robust are linear sketches to adaptive inputs? In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing* (2013), STOC, pp. 121–130.

[21] Harvey, N. J., Nelson, J., and Onak, K. Sketching and streaming entropy via approximation theory. In *49th Annual IEEE Symposium on Foundations of Computer Science* (2008), FOCS, pp. 489–498.

[22] Jayaram, R., and Woodruff, D. P. Data streams with bounded deletions. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (2018), PODS, ACM, pp. 341–354.

[23] Jayaram, R., and Woodruff, D. P. Towards optimal moment estimation in streaming and distributed models. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques* (2019), APPROX/RANDOM.

[24] Jayram, T. S., and Woodruff, D. P. The data stream space complexity of cascaded norms. In *50th Annual IEEE Symposium on Foundations of Computer Science* (2009), pp. 765–774.

[25] Jayram, T. S., and Woodruff, D. P. Optimal bounds for johnson-lindenstrauss transforms and streaming problems with subconstant error. *ACM Transactions on Algorithms (TALG) 9*, 3 (2013), 26.

[26] Kamath, A., Price, E., and Woodruff, D. P. New bounds for L2 heavy hitters in insertion-only streams, 2019.

[27] Kane, D. M., Nelson, J., and Woodruff, D. P. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms* (2010), SODA, pp. 1161–1178.

[28] Kane, D. M., Nelson, J., and Woodruff, D. P. An optimal algorithm for the distinct elements problem. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (2010), PODS, ACM, pp. 41–52.

[29] Li, Y., and Woodruff, D. P. A tight lower bound for high frequency moment estimation with small error. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, APPROX/RANDOM. Springer, 2013, pp. 623–638.

[30] McCurley, K. S. The discrete logarithm problem. In *Proceedings of Symposia in Applied Mathematics* (1990), vol. 42, pp. 49–74.

[31] Mironov, I., Naor, M., and Segev, G. Sketching in adversarial environments. *SIAM Journal on Computing 40*, 6 (2011), 1845–1870.

[32] Misra, J., and Gries, D. Finding repeated elements. *Science of computer programming 2*, 2 (1982), 143–152.

[33] Muthukrishnan, S. Data Streams: Algorithms and Applications. *Foundations and Trends in Theoretical Computer Science 1*, 2 (2005), 117–236.

[34] Naor, M., and Yogev, E. Bloom filters in adversarial environments. In *Advances in Cryptology - CRYPTO - 35th Annual Cryptology Conference* (2015), pp. 565–584.

[35] Schmidt, J. P., Siegel, A., and Srinivasan, A. Chernoff–Hoeffding bounds for applications with limited independence. *SIAM Journal on Discrete Mathematics 8*, 2 (1995), 223–250.

[36] von zur Gathen, J., and Gerhard, J. *Modern Computer Algebra*, 3 ed. Cambridge University Press, 2013.

[37] Woodruff, D. Optimal space lower bounds for all frequency moments. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms* (2004), pp. 167–175.