Relaxed Models for Adversarial Streaming: The Bounded Interruptions Model and the Advice Model

Menachem Sadigurschi
Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Moshe Shechner
Blavatnik School of Computer Science, Tel Aviv University, Israel

Uri Stemmer
Blavatnik School of Computer Science, Tel Aviv University, Israel
Google Research, Tel Aviv, Israel

Abstract

Streaming algorithms are typically analyzed in the *oblivious* setting, where we assume that the input stream is fixed in advance. Recently, there is a growing interest in designing *adversarially robust* streaming algorithms that must maintain utility even when the input stream is chosen *adaptively and adversarially* as the execution progresses. While several fascinating results are known for the adversarial setting, in general, it comes at a very high cost in terms of the required space. Motivated by this, in this work we set out to explore intermediate models that allow us to interpolate between the oblivious and the adversarial models. Specifically, we put forward the following two models:

- *The bounded interruptions model*, in which we assume that the adversary is only partially adaptive.
- *The advice model*, in which the streaming algorithm may occasionally ask for one bit of advice.

We present both positive and negative results for each of these two models. In particular, we present generic reductions from each of these models to the oblivious model. This allows us to design robust algorithms with significantly improved space complexity compared to what is known in the plain adversarial model.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases streaming, adversarial streaming

Digital Object Identifier 10.4230/LIPIcs.ESA.2023.91

Related Version *Full Version*: https://arxiv.org/abs/2301.09203

Funding This project was partially supported by the Israel Science Foundation (grant 1871/19) and by Len Blavatnik and the Blavatnik Family foundation.

1 Introduction

Streaming algorithms are algorithms for processing data streams in which the input is presented as a sequence of items. Generally speaking, these algorithms have access to limited memory, significantly smaller than what is needed to store the entire data stream. This field was formalized by Alon, Matias, and Szegedy [3], and has generated a large body of work that intersects many other fields in computer science. In this work, we focus on streaming algorithms that aim to track a certain function of the input stream, and to continuously report estimates of this function. Formally,

▶ **Definition 1.1** (Informal version of Definition 2.1). *Let X be a finite domain and let $g : X^* \to \mathbb{R}$ be a function that maps every input $\vec{x} \in X^*$ to a real number $g(\vec{x}) \in \mathbb{R}$.*

Let $\mathcal{A}$ be an algorithm that in every round $i \in [m]$ obtains an element $x_i \in X$ and outputs a response $z_i \in \mathbb{R}$. Algorithm $\mathcal{A}$ is said to be an oblivious streaming algorithm *for g with accuracy α, failure probability β, and stream length m, if the following holds for every input sequence $\vec{x} = (x_1, x_2, \ldots, x_m) \in X^m$. Consider an execution of $\mathcal{A}$ on the input stream $\vec{x}$. Then,* $\Pr[\forall i \in [m]$ *we have* $z_i \in (1 \pm \alpha) \cdot g(x_1, \ldots, x_i)] \geq 1 - \beta$, *where the probability is taken over the coins of algorithm $\mathcal{A}$.*

Note that in Definition 1.1, the streaming algorithm is required to succeed (w.h.p.) for every *fixed* input stream. In particular, it is assumed that the choice for the elements in the stream is *independent* from the internal randomness of the streaming algorithm. This assumption, called the *oblivious setting*, is crucial for the correctness of most classical streaming algorithms. In this work, we are interested in a setting where this assumption does not hold, referred to as the *adversarial setting*.

1.1 The (Plain) Adversarial Model

The adversarial streaming model, in various forms, was considered by [20, 12, 13, 1, 2, 15, 8, 7, 16, 24, 6, 4]. The adversarial setting is modeled by a two-player game between a (randomized) `StreamingAlgorithm` and an `Adversary`. At the beginning, we fix a function $g : X^* \to \mathbb{R}$. Throughout the game, the adversary chooses the updates in the stream, and is allowed to *query* the streaming algorithm at T time steps of its choice (referred to as "query times"). Formally, for round $i = 1, 2, \ldots, m$:

1. The `Adversary` chooses an update $x_i \in X$ and a query demand $q_i \in \{0, 1\}$, under the restriction that $\sum_{j=1}^{i} q_j \leq T$.
2. The `StreamingAlgorithm` processes the new update x_i. If $q_i = 1$ then, the `StreamingAlgorithm` outputs a response z_i, which is given to the `Adversary`.

The goal of the `Adversary` is to make the `StreamingAlgorithm` output an incorrect response z_i at some query time i in the stream. Let g be a function defining a streaming problem, and suppose that there is an oblivious streaming algorithm $\mathcal{A}$ for g that uses space s. It is easy to see that g can be solved in the adversarial setting using space $\approx s \cdot T$, by running T copies of $\mathcal{A}$ and using each copy for at most one query. The question is if we can do better. Indeed, Hassidim et al. [16] showed the following result.

▶ **Theorem 1.2** ([16], informal). *If there is an* oblivious *streaming algorithm for a function g that uses space s, then there is an* adversarially robust *streaming algorithm for g supporting T queries using space $\approx \sqrt{T} \cdot s$.*

Note that when the number of queries T is large, this construction incurs a large space blowup. One way for coping with this is to assume additional restrictions on the function g or on the input stream. Indeed, starting with Ben-Eliezer et al. [7], most of the positive results on adversarial streaming assumed that the input stream is restricted to have a small *flip-number*, defined as follows.

▶ **Definition 1.3** (Flip number [7]). *The (α, m)-flip number of an input stream $\vec{x}$ w.r.t. a function g, denoted as $\lambda_{\alpha,m}(\vec{x}, g)$, or simply λ, is the maximal number of times that the value of g changes (increases or decreases) by a factor of $(1 + \alpha)$ during the stream $\vec{x}$.*

Starting from [7], the prior works of [7, 16, 24, 4] presented generic constructions that transform an oblivious streaming algorithm with space s into an adversarially robust streaming algorithm with space $\approx s \cdot \text{poly}(\lambda)$. That is, under the assumption that the flip-number is bounded, these prior works can even support $T = m$ queries. This is useful since the

parameter λ is known to be small for many interesting streaming problems in the *insertion-only* model (where there are no deletions in the stream). However, in general it can be as big as $\Theta(m)$, in which case the transformations of [7, 16, 24, 4] come at a very high cost in terms of space. To summarize this discussion, current reductions from the adversarial to the oblivious setting are useful either when the number of queries T is small, or when the flip-number is small.

1.2 Our Results

One criticism of the adversarial model is that it is (perhaps) too pessimistic. Indeed, there could be many scenarios that do not fall into the oblivious model, but are still quite far from an "adversarial" setting. Motivated by this, in this work, we set out to explore intermediate models that allow us to interpolate between the oblivious model and the adversarial model. Specifically, we study two such models, which we call the *bounded interruptions model* and the *advice model*.

1.2.1 Adversarial Streaming with Bounded Interruptions (ASBI)

Recall that in the plain adversarial model, the adversary is fully adaptive in the sense that it does not commit to the ith update before time i. We consider a refinement of this setting in which the adversary is only *partially* adaptive. The game begins with the adversary specifying a complete input stream. Throughout the execution, the adversary (who sees the outputs of the streaming algorithm) can adaptively decide to *interrupt* and to replace the suffix of the stream (which has not yet been processed by the streaming algorithm). Formally, we define the following *ASBI game* between a `StreamingAlgorithm` and an `Adversary`.

Parameters: m denotes the length of the stream, T denotes the number of allowed queries, and R denotes the number of allowed interruptions.

1. The `Adversary` chooses a stream $\vec{x} = (x_1, x_2, \ldots, x_m) \in X^m$.
2. For round $i = 1, 2, \ldots, m$
   a. The `Adversary` chooses a query demand $q_i \in \{0, 1\}$ and an interruption demand $d_i \in \{0, 1\}$, under the restriction that $\sum_{j=1}^{i} q_j \leq T$ and $\sum_{j=1}^{i} d_j \leq R$.
   b. If $d_i = 1$ then the adversary outputs a new stream suffix $(x'_i, \ldots, x'_m)$ and we override $(x_i, \ldots, x_m) \leftarrow (x'_i, \ldots, x'_m)$.
   c. The `StreamingAlgorithm` obtains the update x_i. If $q_i = 1$ then, the `StreamingAlgorithm` outputs a response z_i, which is given to the `Adversary`.

That is, the adversary sees all of the outputs given by the streaming algorithm (in query times), and adaptively decides on R places in which it "interrupts" and arbitrarily modifies the rest of the stream. Importantly, the streaming algorithm "does not know" when interruptions occur. The ASBI model limits the adaptivity of the adversary in a natural way, capturing settings in which the number of "adaptivity rounds" can be bounded. It gives us an intuitive interpolation between the oblivious setting (in which $R = 0$) and the full adversarial setting (obtained by setting $R = T$).[1]

We show the following generic result.

[1] As described, the plain adversarial setting is obtained by setting $R = m$. However, an easy argument shows that setting $R > T$ does not give more power to the adversary over the case of $R = T$.

▶ **Theorem 1.4** (informal version of Theorem 3.1)**.** *If there exists an oblivious streaming algorithm for a function $g : X^* \to \mathbb{R}$ using space s then for every $1 \leq R \leq T$ there exists an adversarially robust streaming algorithm for g in the ASBI model that answers T queries and resists R interruptions using space $\approx \min\left\{R, \sqrt{T}\right\} \cdot s$.*

To obtain this result, we build on the *sketch switching* technique introduced by Ben-Eliezer et al. [7]. Intuitively, we maintain $2R$ copies of an oblivious streaming algorithm $\mathcal{A}$, where in every given moment exactly two of these copies are designated as "active". As long as the two active copies produce (roughly) the same estimates, they remain as the "active" copies, and we use their estimates as our response. Once they disagree, we discard them both (never to be used again) and designate two (fresh) copies as "active". In Section 3 we show that this construction can be formalized to obtain Theorem 1.4.

▶ **Remark 1.5.** *While our method resembles the sketch switching technique of [7], their technique is tailored to the setting where the flip-number is small (i.e., the value of the target function does not "jump" too many times throughout the stream), and is not directly applicable to the ASBI model. Specifically, in [7] there is only* one *active copy, which is "swithched" the moment its response differs significantly from* the previously released estimate*. This is useful under the assumption that the flip-number is small, because the flip-number bounds the number of switches. However, in the ASBI model we do not assume anything about the flip-number of the stream, and the value of the target function can be completely different at different query times.*

For example, the following is a direct application of Theorem 1.4 for F_2 estimation (i.e., estimating the second moment of the frequency vector of the input stream).

▶ **Theorem 1.6** (F_2 estimation in the ASBI model, informal)**.** *Let $1 \leq R \leq T$. There exists an adversarially robust F_2 estimation algorithm in the ASBI model that guarantees α-accuracy (w.h.p.) for T queries while resisting R interruptions using space $\approx \min\left\{R, \sqrt{T}\right\}/\alpha^2$.*

This improves the state-of-the-art for turnstile streams in the plain adversarial model (from [16]) whenever $R \lesssim \sqrt{T}$.

A negative result for the ASBI model. Note that the space blowup of our construction from Theorem 1.4 grows linearly with the number of interruptions R. Recall that in the full adversarial model (where $R = T$ for T queries) it is known that a space blowup of $\sqrt{T}$ suffices (see Theorem 1.2). Thus, one might guess that the correct dependence in R in the ASBI model should be $\sqrt{R}$. However, we show that this is generally not the case. Specifically, we show that there exists a streaming problem that can easily be solved in the oblivious setting with small space, but necessitates space linear in R in the ASBI model, provided that the number of queries is large enough (quadratic in R). This shows that our upper bound stated in Theorem 1.4 is (nearly) tight in general. Formally,

▶ **Theorem 1.7** (informal version of Corollary 3.8)**.** *For every $R \leq T$ there exists a streaming problem that requires at least $\Omega\left(\min\left\{R, \sqrt{T}\right\}\right)$ space to be solved in the ASBI model with R interruptions and T queries, but can be solved in the oblivious setting using space* $\mathrm{polylog}(T)$.

1.2.2 Adversarial Streaming with Advice (ASA)

We put forward another intermediate model, in which the streaming algorithm may occasionally ask for one bit of *advice* throughout the execution. Formally, we consider the following game, referred to as the ASA game, between the `StreamingAlgorithm` and an `Adversary`.

Parameters: m denotes the length of the stream, $T \leq m$ denotes the number of allowed queries, and $\eta \in \mathbb{N}$ is a parameter controlling the query/advice rate.

For round $i = 1, 2, \ldots, m$:

1. The `Adversary` chooses an update $x_i \in X$ and a query demand $q_i \in \{0, 1\}$, under the restriction that $\sum_{j=1}^{i} q_j \leq T$.
2. The `StreamingAlgorithm` processes the new update x_i.
3. If $q_i = 1$ then
   a. The `StreamingAlgorithm` outputs a response z_i, which is given to the `Adversary`.
   b. If $\left(\sum_{j=1}^{i} q_j\right) \bmod \eta = 0$ then the `StreamingAlgorithm` specifies a predicate $p_i : X^* \to \{0, 1\}$, and obtains $p_i(x_1, x_2, \ldots, x_i)$.

That is, in the ASA model the adversary is allowed a total of T queries, and once every η queries the streaming algorithm is allowed to obtain one bit of advice, computed as a predicate of the items in the stream so far.

Unlike our ASBI model, which (we believe) has a strong algorithmic motivation, the main motivation to study the ASA model is theoretical. It gives us an intuitive way to measure the amount of *additional information* that the streaming algorithm needs in order to maintain utility in the adversarial setting.

▶ **Remark 1.8.** *Even though the main motivation for the ASA model is theoretical, it could also be interesting from an algorithmic standpoint in the following context. Consider a streaming setting in which the input stream is fed into both a (low space) streaming algorithm $\mathcal{A}$ and to a server $\mathcal{S}$. The server has large space and can store all the input stream (and, therefore, can in principle solve the streaming problem itself). However, suppose that the server has some communication bottleneck and is busy serving many other tasks in parallel. Hence we would like to delegate as much of the communication as possible to the "cheap" (low space) streaming algorithm $\mathcal{A}$. The ASA model allows for such a delegation, in the sense that the streaming algorithm handles most of the queries itself, and only once every η queries it asks for one bit of advice from the server.*

We show the following generic result.

▶ **Theorem 1.9** (informal version of Theorem 4.2)**.** *If there exists an oblivious* linear *streaming algorithm for a function $g : X^* \to \mathbb{R}$ with space s, then for every $\eta \in \mathbb{N}$ there exists an adversarially robust streaming algorithm for g in the ASA model with query/advice rate η using space $\approx \eta \cdot s^2$.*

To obtain this result, we rely on a technique introduced by Hassidim et al. [16] which uses *differential privacy* [11] to protect not the input data, but rather the internal randomness of the streaming algorithm. Intuitively, this allows us to make sure that the "robustness" of our algorithm deteriorates *slower* than the advice rate, which allows us to obtain an advice-robustness tradeoff.

Note that the space complexity of the algorithm from Theorem 1.9 does not depend polynomially on the number of queries T. For example, the following is a direct application of this theorem in the context of F_2 estimation.

▶ **Theorem 1.10** (F_2 estimation in the ASA model, informal)**.** *Let $\eta \in \mathbb{N}$. There exists an adversarially robust F_2 estimation algorithm in the ASA model with query/advice rate η that guarantees α-accuracy (w.h.p.) using space $\tilde{O}\left(\eta/\alpha^4\right)$.*

▶ **Remark 1.11.** *We stress that there is a formal sense in which the ASA model is "between" the oblivious and the (plain) adversarial models. Clearly, the ASA model is easier than the plain adversarial model, as we can simply ignore the advice bits. On the other hand, a simple argument shows that the ASA model (with any $\eta > 1$) is qualitatively harder than the oblivious setting. To see this, let $\mathcal{A}$ be an algorithm in the ASA model for a function g with query/advice rate $\eta > 1$. Then $\mathcal{A}$ can be transformed into the following oblivious algorithm $\mathcal{A}_{\text{oblivious}}$ for g (that returns an estimate in every time step without getting any advice):*

An oblivious algorithm $\mathcal{A}_{\text{oblivious}}$

1. *Instantiate algorithm $\mathcal{A}$ (which expects to operate in the ASA model).*
2. *In every time $i \in [m]$:*
   a. *Obtain an update $x_i \in X$.*
   b. *Duplicate $\mathcal{A}$ (with its internal state) into a shadow copy $\mathcal{A}^{\text{shadow}}$.*
   c. *Feed the update $(x_i, 0)$ to $\mathcal{A}$ and the update $(x_i, 1)$ to $\mathcal{A}^{\text{shadow}}$, and obtain an answer z_i from the shadow copy. Here we use the notation that (x, q) means an update x with a query demand q. Note that we only query the shadow copy.*
   d. *Output z_i and erase the shadow copy from memory.*

As we "rewind" $\mathcal{A}$ after every query, it is never expected to issue an advice-request and so $\mathcal{A}_{\text{oblivious}}$ never issues an advice-request as well. Furthermore, a simple argument shows that $\mathcal{A}_{\text{oblivious}}$ maintains utility in the oblivious setting.[2]

▶ **Remark 1.12.** *Our construction has the benefit that the predicates specified throughout the interaction are "simple" in the sense that every single one of them can be computed in a streaming fashion. That is, given the predicate p_i, the bit $p_i(x_1, x_2, \ldots, x_i)$ can be computed using small space with one pass over $x_1, x_2, \ldots, x_i$.*

A negative result for the ASA model. Theorem 1.9 shows a strong positive result in the ASA model for streaming problems that are defined by real valued functions, presenting space complexity that does not depend directly on the number of queries T. We compliment this result with a negative result for a simple streaming problem which is *not* defined by a real valued function. Specifically, we consider (a variant of) the well-studied ℓ_0-sampling problem, where the streaming algorithm must return an (almost) uniformly random element from the set of non-deleted elements. It is known that the ℓ_0-sampling problem is easy in the oblivious setting (see e.g. [17]) and hard in the plain adversarial setting (see e.g. [1]). Using a simple counting argument, we show that the ℓ_0-sampling problem remains hard also in the ASA model *even if the query/advice rate is 1*, i.e., even if the streaming algorithm gets an advice bit for *every* query. Formally,

▶ **Theorem 1.13** (informal version of Theorem 4.5)**.** *Every algorithm for solving the ℓ_0-sampling problem in the ASA model with T queries must use space $\Omega(T)$. Furthermore, this holds even when $\eta = 1$, that is, even if the algorithm gets an advice bit after* every *query.*

[2] To see this, fix an input stream $\vec{x} = (x_1, x_2, \ldots, x_m)$, and fix $j \in [m]$. Note that the distribution of the output given by $\mathcal{A}_{\text{oblivious}}$ in time j when running on $\vec{x}$ is identical to the outcome distribution of $\mathcal{A}$ when running on the stream $((x_1, 0), \ldots, (x_{j-1}, 0), (x_j, 1))$, which must be accurate w.h.p. by the utility guarantees of $\mathcal{A}$ (since there is only 1 query in this alternative stream, then $\mathcal{A}$ gets no advice when running on it). The claim now follows by a union bound over the query times.

1.3 Additional Related Works

The adversarial streaming model (in a setting similar to ours) dates back to at least [1], who studied it implicitly and showed an impossibility result for robust ℓ_0 sampling in sublinear memory. The adversarial streaming model was then formalized explicitly by [15], who showed strong impossibility results for linear sketches. A recent line of work, starting with [7] and continuing with [16, 24, 4, 6] showed *positive* results (i.e., *robust algorithms*) for many problems of interest, under the assumption that the *flip-number* of the stream is bounded. On the negative side, [7] also presented an attack with $O(n)$ number of adaptive rounds on a variant of the AMS sketch, where n is the size of the domain. Later, [19] presented a streaming problem that can be solved in the oblivious setting with polylogarithmic space, but requires polynomial space in the adversarial setting. Thus, the result of [19] separates the oblivious model from the (plain) adversarial model. More recently, [9] and [10] presented attacks on a concrete algorithm, namely `CountSketch`, with a number of queries that is *linear* in the space of the algorithm and while using only a single round of adaptivity. Following [16], the idea of using differential privacy to protect the internal randomness of interactive algorithms was used in additional settings, for example in the context of dynamic algorithms [5] and learning with experts [23].

2 Preliminaries

In this work we consider streaming problems which are defined by a real valued function (where the goal is to approximate the value of this function) as well as streaming problems that define a set of valid solutions and the goal is to return one of the valid solutions. The following definition unifies these two objectives for the oblivious setting.

▶ **Definition 2.1** (Oblivious streaming). *Let X be a finite domain and let $g : X^* \to 2^W$ be a function that maps every input $\vec{x} \in X^*$ to a subset $g(\vec{x}) \subseteq W$ of valid solutions (from some range W).*

Let $\mathcal{A}$ be an algorithm that, for m rounds, obtains an element $x_i \in X$ and outputs a response $z_i \in W$. Algorithm $\mathcal{A}$ is said to be an oblivious streaming algorithm *for g with failure probability β, and stream length m, if the following holds for every input sequence $\vec{x} = (x_1, x_2, \ldots, x_m) \in X^m$. Consider an execution of $\mathcal{A}$ on the input stream $\vec{x}$. Then,*

$$\Pr\left[\forall i \in [m] \text{ we have } z_i \in g(x_1, \ldots, x_i)\right] \geq 1 - \beta,$$

where the probability is taken over the coins of algorithm $\mathcal{A}$.

For example, in the problem of estimating the number of distinct elements in the stream, the function g in the above definition returns the interval $g(x_1, \ldots, x_i) = (1 \pm \alpha) \cdot |\{x_1, \ldots, x_i\}|$, where α is the desired approximation parameter.

3 Adversarial Streaming with Bounded Interruptions (ASBI)

In this section we present our results for the ASBI model, defined in Section 1.2.1. We begin with our generic transformation.

3.1 A Generic Construction for the ASBI Model

We present a generic construction whose space blowup depends only on the number of interruptions R (and not on the number of queries T). We therefore assume in our construction that $T = m$, i.e., that the adversary queries the streaming algorithm on *every* time step. Our construction is given in algorithm `RobustInterruptions`. The following theorem specifies its properties.

Algorithm 1 `RobustInterruptions`.

Input: Parameter $R \in \mathbb{N}^+$ bounding the number of possible interruptions.

Algorithm used: An oblivious streaming algorithm $\mathcal{A}$ with space s, accuracy α, and confidence β.

1. Initialize $2R$ independent instances of algorithm $\mathcal{A}$, denoted as $\mathcal{A}_1^{\text{answer}}, \ldots, \mathcal{A}_R^{\text{answer}}$ and $\mathcal{A}_1^{\text{check}}, \ldots, \mathcal{A}_R^{\text{check}}$. Set $r = 1$.
2. For $i = 1, 2, \ldots, m$:
   a. Obtain the next item in the stream $x_i \in X$.
   b. Feed x_i to *all* of the copies of algorithm $\mathcal{A}$.
   c. Let $z_{r,i}^{\text{answer}}$ and $z_{r,i}^{\text{check}}$ denote the answers returned by $\mathcal{A}_r^{\text{answer}}$ and $\mathcal{A}_r^{\text{check}}$, respectively.
   d. If $z_{r,i}^{\text{answer}} \in (1 \pm 2\alpha) \cdot z_{r,i}^{\text{check}}$ then output $z_{r,i}^{\text{answer}}$. Otherwise, output $z_{r,i}^{\text{check}}$ and set $r \leftarrow r + 1$.
   e. If $r > R$ then FAIL. Otherwise continue to the next iteration.

▶ **Theorem 3.1.** *Fix any function g and fix $\alpha, \beta > 0$. Let $\mathcal{A}$ be an oblivious streaming algorithm for g that uses space s and guarantees accuracy α with failure probability β. Then for all $R \in \mathbb{N}^+$ there exists an adversarially robust streaming algorithm for g that resists R interruptions and guarantees accuracy 5α with failure probability $O(R\beta)$ using space $O(Rs)$.*

▶ **Remark 3.2.** *Recall that Hassidim et al. [16] showed a generic transformation from an oblivious algorithm for a function $g : X^* \to \mathbb{R}$ with space s to an adversarial algorithm (in the plain model) for g, answering T queries with space $\approx \sqrt{T} \cdot s$ (see Theorem 1.2). Therefore, combined with the above theorem, we get that for every such function g and every $1 \leq R \leq T$ there exists an adversarially robust streaming algorithm in the ASBI model that answers T queries and resists R interruptions using space $\approx \min\left\{R, \sqrt{T}\right\} \cdot s$.*

Fix an adversary $\mathcal{B}$ and consider the interaction between algorithm `RobustInterruptions` and the adversary $\mathcal{B}$. For $r \in [R]$, let i_r denote the time step in which z_{r,i_r}^{check} is returned.

▶ **Lemma 3.3.** *Fix $r^* \in [R]$. With probability at least $1 - \beta$, the answers returned by $\mathcal{A}_{r^*}^{\text{check}}$ in times $1, 2, \ldots, i_{r^*}$ are α-accurate. That is, for every $1 \leq i \leq i_{r^*}$ it holds that $z_{r^*,i}^{\text{check}} \in (1 \pm \alpha) \cdot g(x_1, \ldots, x_i)$.*

Proof. For simplicity, we assume that the adversary $\mathcal{B}$ is deterministic (this is without loss of generality by a simple averaging argument). Fix the randomness of all copies of algorithm $\mathcal{A}$, except for $\mathcal{A}_{r^*}^{\text{check}}$. Let $\texttt{RI}_{r^*}$ be a variant of algorithm `RobustInterruptions` which is identical to `RobustInterruptions` until the time step i^* in which r becomes r^*. In times $i \geq i^*$, algorithm $\texttt{RI}_{r^*}$ simply outputs $z_{r^*,i}^{\text{answer}}$, i.e., the answer given by $\mathcal{A}_{r^*}^{\text{answer}}$. Note that $\mathcal{A}_{r^*}^{\text{check}}$ does not exist in algorithm RI_{r^*}.

As we fixed the coins of the copies of $\mathcal{A} \neq \mathcal{A}_{r^*}^{\text{check}}$, the interaction between $\mathcal{B}$ and $\texttt{RI}_{r^*}$ is deterministic. In particular, it generates a single stream $\vec{x}_{r^*}$. By the utility guarantees of algorithm $\mathcal{A}_{r^*}^{\text{check}}$, when run on this stream, then with probability at least $1 - \beta$ it maintains α-accuracy throughout the stream.

The lemma now follows by observing that until time i_{r^*} the stream generated by the interaction between $\mathcal{B}$ and algorithm `RobustInterruptions` is identical to the stream $\vec{x}_{r^*}$. ◀

▶ **Lemma 3.4.** *With probability at least $1 - R\beta$, all of the answers given by `RobustInterruptions` (before returning FAIL) are 5α-accurate.*

Proof. Follows from a union bound over Lemma 3.3, and by Step 2d of `RobustInterruptions`. ◀

▶ **Lemma 3.5.** *Algorithm* `RobustInterruptions` *returns FAIL with probability at most* $2R\beta$.

Proof. Let $j_1, j_2, \ldots, j_R$ denote the time steps in which the adversary conducts interruptions. That is, j_1 is the first time in which the adversary switches the suffix of the stream, j_2 is the second time this happens, and so on. Also let $p_1, p_2, \ldots, p_R$ denote the time steps in which the parameter r increases during the execution of algorithm `RobustInterruptions`. Specifically, p_ℓ is the time i in which r becomes equal to $\ell + 1$. We show that for every $r \in [R]$, with probability at least $1 - 2r\beta$ it holds that $j_r \leq p_r$. (That is, interruptions happen "faster" then r increases.)

The proof is by induction on r. For the base case, $r = 1$, let $\vec{x}_1$ denote the first stream chosen by the adversary. By the utility guarantees of $\mathcal{A}$, with probability at least $1 - 2\beta$ we have that both $\mathcal{A}_1^{\text{answer}}$ and $\mathcal{A}_1^{\text{check}}$ are α-accurate w.r.t. this stream, in which case r does not increase. Thus, with probability at least $1 - 2\beta$ we have $j_1 \leq p_1$.

The inductive step is similar: Fix $r \in [R]$, and suppose that $j_r \leq p_r$, which happens with probability at least $(1 - 2r\beta)$ by the inductive assumption. Let $\vec{x}_r$ denote the last stream specified by the adversary before time p_r. Note that the internal coins of $\mathcal{A}_{r+1}^{\text{answer}}$ and $\mathcal{A}_{r+1}^{\text{check}}$ are independent with this stream. Hence, by the utility guarantees of $\mathcal{A}$, with probability at least $1 - 2\beta$ we have that both $\mathcal{A}_{r+1}^{\text{answer}}$ and $\mathcal{A}_{r+1}^{\text{check}}$ are α-accurate w.r.t. this stream, in which case r does not increase. Overall, with probability at least $1 - 2(r+1)\beta$ we have $j_{r+1} \leq p_{r+1}$.

The lemma now follows by recalling that there are at most R interruptions throughout the execution. Hence, with probability at least $1 - 2R\beta$ it holds that r never increases beyond R, and the algorithm does not fail. ◀

Theorem 3.1 now follows by combining Lemmas 3.3, 3.4, 3.5.

3.2 A Negative Result for the ASBI Model

We show the following negative result.

▶ **Theorem 3.6.** *For every R, there exists a streaming problem over domain of size* $\text{poly}(R)$ *and stream length* $\text{poly}(R)$ *that requires at least* $\Omega(R)$ *space to be solved in the ASBI model with R interruptions and $T = R^2$ queries to within constant accuracy (small enough), but can be solved in the oblivious setting using space* $\text{polylog}(R)$.

We next provide an overview for the proof of this theorem (see the full version of this work for more details). At a high level, Theorem 3.6 follows by strengthening the following negative result of Kaplan et al. [19] for the (plain) adversarial model.

▶ **Theorem 3.7** ([19])**.** *For every T, there exists a streaming problem over domain of size* $\text{poly}(T)$ *and stream length* $\text{poly}(T)$ *that requires at least $\sqrt{T}$ space to be solved in the (plain) adversarial setting with T queries to within constant accuracy (small enough), but can be solved in the oblivious setting using space* $\text{polylog}(T)$.

To obtain their result, Kaplan et al. [19] presented a streaming problem, called the SADA problem, that is easy to solve in the oblivious setting but requires large space to be solved in the adversarial setting. Specifically, they showed a reduction from a hard problem in learning theory (called the *adaptive data analysis (ADA) problem*) to the task of solving the SADA problem in the adversarial setting with small space.

In the ADA problem, the goal is to design a mechanism $\mathcal{A}$ that initially obtains a dataset D consisting of n i.i.d. samples from some unknown distribution $\mathcal{P}$, and then answers k *adaptively chosen queries* w.r.t. $\mathcal{P}$. Importantly, $\mathcal{A}$'s answers must be accurate w.r.t. the underlying distribution $\mathcal{P}$, and not just w.r.t. the empirical dataset D. Hardt, Ullman, and Steinke [14, 22] showed that the ADA problem requires a large sample complexity. Specifically, they showed that every efficient[3] mechanism for this problem must have sample complexity $n \geq \Omega(\sqrt{k})$.

Theorem 3.6 follows by the following two observations regarding the negative result of [19] for the SADA problem, and regarding the underlying hardness result of [14, 22] for the ADA problem:

1. In the hardness results of [14, 22] for the ADA problem, the adversary generates the queries using $O(n)$ rounds of adaptivity, where n is the sample size. In more detail, the adversary poses $O(n^2)$ queries throughout the interaction, but these queries are generated in $O(n)$ bulks where queries in the jth bulk depend only on answers given to queries of *previous* bulks.
2. The reduction of Kaplan et al. [19] from the ADA problem to the SADA problem maintains the number of adaptivity rounds. That is, the reduction of Kaplan et al. [19] transforms an adversary for the ADA problem that generates the queries in ℓ bulks into an adversary for the SADA problem that uses ℓ interruptions.

The following is an immediate corollary of Theorem 3.6.

▶ **Corollary 3.8.** *For every $R \leq T$, there exists a streaming problem over domain of size* $\text{poly}(T)$ *and stream length* $\text{poly}(T)$ *that requires at least* $\Omega\left(\min\left\{R, \sqrt{T}\right\}\right)$ *space to be solved in the ASBI model with R interruptions and T queries to within constant accuracy (small enough), but can be solved in the oblivious setting using space* $\text{polylog}(T)$.

Observe that this (nearly) matches our upper bound stated in Theorem 1.4. We remark that Theorem 3.6 and Corollary 3.8 hold even in a model where the streaming algorithm is strengthen and gets an indication during each interruption round.

4 Adversarial Streaming with Advice (ASA)

In this section we present our results for the ASA model, defined in Section 1.2.2. We begin with our generic transformation.

4.1 A Generic Construction for the ASA Model

Our generic construction for the ASA model transforms an oblivious and *linear* streaming algorithm $\mathcal{A}$ into a robust streaming algorithm in the ASA model. The linearity property that we need is the following. Suppose that three copies of $\mathcal{A}$, call them $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$, are instantiated with the *same* internal randomness r, and suppose that $\mathcal{A}_1$ processes a stream $\vec{x}_1$ and that $\mathcal{A}_2$ processes a stream $\vec{x}_2$ and that $\mathcal{A}_3$ processes the stream $\vec{x}_1 \circ \vec{x}_2$ (where the operator $\circ$ stands for concatenation). Then there is an operation, denote it as "+", that allows us to obtain an internal state $(\mathcal{A}_1 + \mathcal{A}_2)$ that is identical to the internal state of $\mathcal{A}_3$. Many classical streaming algorithms have this property (for example, the classical AMS sketch for F_2 has this property [3]). Formally,

[3] The results of [14, 22] hold for all computationally efficient mechanisms, or alternatively, for a class of unbounded mechanisms which they call *natural* mechanisms.

Algorithm 2 `RobustAdvice`(β, m, η).

Input: Parameters: β is the failure probability, m is the length of input stream and η is the advice query cycle.

Algorithm used: An oblivious linear streaming algorithm $\mathcal{A}$ with space s for α accuracy.

Constants calculation:

1. $k = \Omega(\eta s \log(m/\beta) \log^2(m/(\beta\alpha)))$ is the number of instances of each of the sets "active", "next" and "shadow".
2. $\varepsilon_0 = \frac{\varepsilon}{\sqrt{8\eta k s \ln(1/\delta)}}$ is the privacy parameter of `PrivateMed` executions, where $\varepsilon = 1/100$, $\delta = O(\beta/m)$.

- -

1. Initialize k independent instances $\mathcal{A}_1^{\text{active}}, \ldots, \mathcal{A}_k^{\text{active}}$ of algorithm $\mathcal{A}$.
2. REPEAT (outer loop)
   a. Initialize k independent instances $\mathcal{A}_1^{\text{next}}, \ldots, \mathcal{A}_k^{\text{next}}$ of algorithm $\mathcal{A}$.
   b. Let $\mathcal{A}_1^{\text{shadow}}, \ldots, \mathcal{A}_k^{\text{shadow}}$ be duplicated copies of $\mathcal{A}_1^{\text{next}}, \ldots, \mathcal{A}_k^{\text{next}}$, where each $\mathcal{A}_i^{\text{shadow}}$ is initiated with the same randomness as $\mathcal{A}_i^{\text{next}}$.
   c. Denote the current time step as t. (That is, so far we have seen t updates in the stream.)
   d. REPEAT (inner loop)
      i. Receive next update x_t and a query demand $q_t \in \{0, 1\}$.
      ii. Insert update x_t into each of $\mathcal{A}_1^{\text{active}}, \mathcal{A}_1^{\text{next}}, \ldots, \mathcal{A}_k^{\text{active}}, \mathcal{A}_k^{\text{next}}$.
      iii. If $q_t = 1$ then:
         - Query $\mathcal{A}_1^{\text{active}}, \mathcal{A}_2^{\text{active}}, \ldots, \mathcal{A}_k^{\text{active}}$ and obtain answers $y_{t,1}, y_{t,2}, \ldots, y_{t,k}$
         - Output $z_t \leftarrow$ `PrivateMed`$(y_{t,1}, y_{t,2}, \ldots, y_{t,k})$ with privacy parameter ε_0.
         - If $\left(\sum_{\ell=1}^{t} q_\ell\right) \bmod \eta = 0$ then define the predicate p_t that given a (prefix of a) stream $\vec{x}$ returns the next bit in the inner state of $(\mathcal{A}_1^{\text{shadow}}, \mathcal{A}_2^{\text{shadow}}, \ldots, \mathcal{A}_k^{\text{shadow}})$ after processing the first t updates in $\vec{x}$. Update the corresponding bit in the state of the corresponding $\mathcal{A}_i^{\text{shadow}}$.
         - If $\left(\sum_{\ell=1}^{t} q_\ell\right) \bmod \eta k s = 0$ then EXIT inner loop. Otherwise, CONTINUE inner loop.
   e. For $i \in [k]$ let $\mathcal{A}_i^{\text{active}}.\mathcal{S}_v \leftarrow \mathcal{A}_i^{\text{next}}.\mathcal{S}_v + \mathcal{A}_i^{\text{shadow}}.\mathcal{S}_v$ and let $\mathcal{A}_i^{\text{active}}.\mathcal{S}_R \leftarrow \mathcal{A}_i^{\text{next}}.\mathcal{S}_R$.

▶ **Definition 4.1** (Linear state algorithm). *Let $\mathcal{A}$ be an algorithm with three segments of memory state. The first segment, denoted as $\mathcal{S}_R$, is randomized in the beginning of the execution and then remains fixed. The second segment, denoted as $\mathcal{S}_v$, is an encoding of a vector in $\mathbb{R}^d$. The third segment, denoted as $\mathcal{S}_c$, is the rest of its memory space and is used for other computations. Then, $\mathcal{A}$ is* linear state *w.r.t. its input stream if for any two streams $\vec{x}_1 = (x_1, \ldots, x_l) \in X^l$, $\vec{x}_2 = (x_1, \ldots, x_p) \in X^p$ with lengths of $l, p \in \mathbb{N}$ and three different executions of $\mathcal{A}$ with the same randomized state ($\mathcal{S}_R$) the following holds:*

$$\mathcal{A}(\vec{x}_1 \circ \vec{x}_2).\mathcal{S}_v = \mathcal{A}(\vec{x}_1).\mathcal{S}_v + \mathcal{A}(\vec{x}_2).\mathcal{S}_v$$

Where $\mathcal{A}(\vec{x}).\mathcal{S}_v$ is the encoded vector $v \in \mathbb{R}^d$ resulting from the input stream $\vec{x}$ encoded in the corresponding memory state.

Our construction is given in algorithm `RobustAdvice`, and its properties are stated in the following theorem.[4]. The analysis is deferred to the the full version of this work [21].

▶ **Theorem 4.2.** *Fix any real valued function g and fix $\alpha, \beta > 0$ and $\eta \in \mathbb{N}$. Let $\mathcal{A}$ be an oblivious linear-state streaming algorithm for g that uses space s and guarantees accuracy α with failure probability $1/10$. Then there exists an adversarially robust streaming algorithm for g in the ASA model with query/advice rate η, accuracy α, and failure probability β using space $O(\eta s^2 \log(m/\beta) \log^2(m/(\beta\alpha)))$.*

4.2 A Negative Result for the ASA Model

In this section we show that ℓ_0-sampling, a classical streaming problem, cannot be solved with sublinear space in the adversarial setting with advice. Consider a turnstile stream $\vec{u} = (u_1, \ldots, u_m)$ where each $u_i = (a_i, \Delta_i) \in [n] \times \{\pm 1\}$. A β-error ℓ_0-sampler returns with probability at least $1 - \beta$ an (almost) uniformly random element from

$$\text{support}(u_1, \ldots, u_m) = \left\{ a \in [n] : \sum_{i:a_i=a} \Delta_i \neq 0 \right\},$$

provided that this support is not empty. The next theorem, due to Jowhari et al. [18], shows that ℓ_0-sampling is easy in the oblivious setting.

▶ **Theorem 4.3** ([18])**.** *There is a streaming algorithm with storage $O\left(\log^2(n) \log(\frac{1}{\beta})\right)$ bits, that with probability at most β reports FAIL, with probability at most $1/n^2$ reports an arbitrary answer, and in all other cases produces a uniform sample from* $\text{support}(\vec{u})$.

Nevertheless, as we next show, this is a hard problem in the ASA setting. In fact, our negative result even holds for a simpler variant of the ℓ_0-sampling problem, in which the algorithm is allowed to return an *arbitrary* element, rather than a random element. Formally,

▶ **Definition 4.4** (The J_0 problem)**.** *Let X be a finite domain and let $\mathcal{A}$ be an algorithm that operates on a stream of updates $(u_1, \ldots, u_m) \in (X \times \{\pm 1\})$, given to $\mathcal{A}$ one by one. Algorithm $\mathcal{A}$ solves the J_0 problem with failure probability β if, except with probability at most β, whenever $\mathcal{A}$ is queried it outputs an element with non-zero frequency w.r.t. the current stream. That is, if $\mathcal{A}$ is queried in time i then it should output an element from* $\text{support}(u_1, \ldots, u_i)$.

▶ **Theorem 4.5.** *Let X be a finite domain and let T be such that $|X| = \Omega(T)$ (large enough). Let $\mathcal{A}$ be an algorithm for solving the J_0 problem over X in the adversarial setting with advice with T queries and with failure probability at most $3/4$. Then $\mathcal{A}$ uses space $\Omega(T)$. Furthermore, this holds also when $\eta = 1$, that is, even if algorithm $\mathcal{A}$ gets an advice after* every *query.*

Proof. Let $\mathcal{A}$ be an algorithm for the J_0 problem with T queries over domain X in the ASA setting with failure probability at most $3/4$. Consider the following thought experiment.

[4] We assume that the estimates given by the oblivious algorithm $\mathcal{A}$ are in the range $[-m^c, -1/m^c] \cup \{0\} \cup [1/m^c, m^c]$ for some constant c, and are rounded to the nearest power of $(1 + \alpha)$. See the full version of this work for more details [21].

Input: $Y \subseteq X$ of size $|Y| = T$
1. For every $x \in Y$, feed algorithm $\mathcal{A}$ the update $(x, 1)$.
2. Initiate $\hat{Y} = \emptyset$.
3. Repeat T times:
   a. Query $\mathcal{A}$ and obtain an outcome $x \in X$
   b. If $\mathcal{A}$ requests an advice then give it a random bit b.
   c. Add x to $\hat{Y}$
   d. Feed the update $(x, -1)$ to $\mathcal{A}$
4. Output $\hat{Y}$.

We say that the thought experiment *succeeds* if $\hat{Y} = Y$. By the assumption on $\mathcal{A}$, for every input Y, our thought experiment succeeds with probability at least $2^{-T}/4$. This is because if all of the bits of advice are correct then $\mathcal{A}$ succeeds with probability at least $1/4$, and the advice bits are all correct with probability at least 2^{-T}. Hence, there must exist a fixture of $\mathcal{A}$'s coins and a fixture of an advice string $\vec{b}$ for which our thought experiments succeeds on at least $2^{-T}/4$ fraction of the possible inputs Y.[5]

That is, after fixing $\mathcal{A}$'s coins and the advice string $\vec{b}$ as above, there is a subset of inputs $\mathfrak{B}$ of size $|\mathfrak{B}| \geq \frac{2^{-T}}{4}\binom{|X|}{T}$ such that for every $Y \in \mathfrak{B}$, when executed on Y, our thought experiment outputs $\hat{Y} = Y$. Finally, note that the inner state of algorithm $\mathcal{A}$ at the end of Step 1 *determines* the outcome of our thought experiment. Hence, as there are at least $\frac{2^{-T}}{4}\binom{|X|}{T}$ different outputs, there must be at least $\frac{2^{-T}}{4}\binom{|X|}{T}$ possible different inner states for algorithm $\mathcal{A}$, meaning that its space complexity (in bits) is at least $\log\left(\frac{2^{-T}}{4}\binom{|X|}{T}\right)$, which is more than T provided that $|X| = \Omega(T)$ (large enough). ◀

References

1 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *SODA*, pages 459–467, 2012. doi:10.1137/1.9781611973099.40.

2 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *PODS*, pages 5–14, 2012. doi:10.1145/2213556.2213560.

3 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999. doi:10.1006/jcss.1997.1545.

4 Idan Attias, Edith Cohen, Moshe Shechner, and Uri Stemmer. A framework for adversarial streaming via differential privacy and difference estimators. In *ITCS*, volume 251 of *LIPIcs*, pages 8:1–8:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.

5 Amos Beimel, Haim Kaplan, Yishay Mansour, Kobbi Nissim, Thatchaphol Saranurak, and Uri Stemmer. Dynamic algorithms against an adaptive adversary: generic constructions and lower bounds. In *STOC*, pages 1671–1684. ACM, 2022.

6 Omri Ben-Eliezer, Talya Eden, and Krzysztof Onak. Adversarially robust streaming via dense-sparse trade-offs. In *SOSA@SODA*, pages 214–227, 2022.

7 Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. *J. ACM*, 69(2):17:1–17:33, 2022.

[5] Otherwise, consider sampling an input Y uniformly. We have that $\frac{2^{-T}}{4} \leq \Pr_{r,\vec{b},Y}[\mathcal{A}_{r,\vec{b}}(Y) \text{ succeeds}] = \sum_{r,\vec{b}} \Pr[r,\vec{b}] \cdot \Pr_Y[\mathcal{A}_{r,\vec{b}}(Y) \text{ succeeds}] < \sum_{r,\vec{b}} \Pr[r,\vec{b}] \cdot \frac{2^{-T}}{4} = \frac{2^{-T}}{4}$, which is a contradiction. Here r denotes the randomness of $\mathcal{A}$ and $\vec{b}$ is the advice string.

8 Omri Ben-Eliezer and Eylon Yogev. The adversarial robustness of sampling. In *PODS*, pages 49–62, 2020.

9 Edith Cohen, Xin Lyu, Jelani Nelson, Tamás Sarlós, Moshe Shechner, and Uri Stemmer. On the robustness of countsketch to adaptive inputs. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 4112–4140. PMLR, 2022.

10 Edith Cohen, Jelani Nelson, Tamas Sarlos, and Uri Stemmer. Tricking the hashing trick: A tight lower bound on the robustness of countsketch to adaptive inputs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6):7235–7243, 2023.

11 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

12 A. C. Gilbert, B. Hemenway, A. Rudra, M. J. Strauss, and M. Wootters. Recovering simple signals. In *Information Theory and Applications Workshop (ITA)*, pages 382–391, 2012.

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

14 Moritz Hardt and Jonathan Ullman. Preventing false discovery in interactive data analysis is hard. In *FOCS*. IEEE, october 19-21 2014.

15 Moritz Hardt and David P. Woodruff. How robust are linear sketches to adaptive inputs? In *STOC*, pages 121–130, 2013.

16 Avinatan Hassidim, Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. Adversarially robust streaming algorithms via differential privacy. *J. ACM*, 2022. doi:10.1145/3556972.

17 Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *PODS*, pages 49–58. ACM, 2011.

18 Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 49–58, 2011.

19 Haim Kaplan, Yishay Mansour, Kobbi Nissim, and Uri Stemmer. Separating adaptive streaming from oblivious streaming using the bounded storage model. In *CRYPTO*, pages 94–121, 2021.

20 Ilya Mironov, Moni Naor, and Gil Segev. Sketching in adversarial environments. *SIAM J. Comput.*, 40(6):1845–1870, 2011. doi:10.1137/080733772.

21 Menachem Sadigurschi, Moshe Shechner, and Uri Stemmer. Relaxed models for adversarial streaming: The bounded interruptions model and the advice model, 2023. arXiv:2301.09203.

22 Thomas Steinke and Jonathan R. Ullman. Interactive fingerprinting codes and the hardness of preventing false discovery. In *2016 Information Theory and Applications Workshop, ITA 2016, La Jolla, CA, USA, January 31 – February 5, 2016*, pages 1–41. IEEE, 2016. doi:10.1109/ITA.2016.7888199.

23 David P. Woodruff, Fred Zhang, and Samson Zhou. Streaming algorithms for learning with experts: Deterministic versus robust. *CoRR*, abs/2303.01709, 2023. arXiv:2303.01709.

24 David P. Woodruff and Samson Zhou. Tight bounds for adversarially robust streams and sliding windows via difference estimators. In *FOCS*, pages 1183–1196, 2022.