Separating Adaptive Streaming from Oblivious Streaming using the Bounded Storage Model

Haim Kaplan[1,2⋆], Yishay Mansour[1,2⋆⋆], Kobbi Nissim[3⋆⋆⋆], and Uri Stemmer[2,4†]

[1] Tel Aviv University, Tel Aviv, Israel
[2] Google Research, Tel Aviv, Israel
[3] Georgetown University, Washington, D.C., USA
[4] Ben-Gurion University of the Negev, Be'er-Sheva, Israel

Abstract. Streaming algorithms are algorithms for processing large data streams, using only a limited amount of memory. Classical streaming algorithms typically work under the assumption that the input stream is chosen independently from the internal state of the algorithm. Algorithms that utilize this assumption are called *oblivious* algorithms. Recently, there is a growing interest in studying streaming algorithms that maintain utility also when the input stream is chosen by an *adaptive adversary*, possibly as a function of previous estimates given by the streaming algorithm. Such streaming algorithms are said to be *adversarially-robust*.

By combining techniques from *learning theory* with cryptographic tools from the *bounded storage model*, we separate the oblivious streaming model from the adversarially-robust streaming model. Specifically, we present a streaming problem for which every adversarially-robust streaming algorithm must use polynomial space, while there exists a classical (oblivious) streaming algorithm that uses only polylogarithmic space. This is the first general separation between the capabilities of these two models, resolving one of the central open questions in adversarial robust streaming.

Keywords: Adversarially-robust streaming · Bounded storage model · Separation from oblivious streaming.

⋆ Partially supported by the Israel Science Foundation (grant 1595/19), the German-Israeli Foundation (grant 1367/2017), and by the Blavatnik Family Foundation. haimk@tau.ac.il

⋆⋆ This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 882396), by the Israel Science Foundation (grant number 993/17) and the Yandex Initiative for Machine Learning at Tel Aviv University. mansour.yishay@gmail.com

⋆⋆⋆ Supported by NSF grant No. 1565387 TWC: Large: Collaborative: Computing Over Distributed Sensitive Data and by a gift to Georgetown University, the Data Co-Ops project. kobbi.nissim@georgetown.edu

† Partially Supported by the Israel Science Foundation (grant 1871/19) and by the Cyber Security Research Center at Ben-Gurion University of the Negev. u@uri.co.il

1 Introduction

Consider a scenario in which data items are being generated one by one, e.g., IP traffic monitoring or web searches. Generally speaking, streaming algorithms aim to process such data streams while using only a limited amount of memory, significantly smaller than what is needed to store the entire data stream. Streaming algorithms have become a central and crucial tool for the analysis of massive datasets.

A typical assumption when designing and analyzing streaming algorithms is that the entire stream is *fixed* in advance (and is just provided to the streaming algorithm one item at a time), or at least that the choice of the items in the stream is *independent* of the internal state (and coin tosses) of the streaming algorithm. We refer to this setting as the *oblivious* setting. Recently, there has been a growing interest in streaming algorithms that maintain utility even when the choice of stream items depends on previous answers given by the streaming algorithm, and can hence depend on the internal state of the algorithm [21, 13, 14, 1, 2, 16, 7, 6, 18, 26]. Such streaming algorithms are said to be *adversarially robust*.

Hardt and Woodruff [16] presented a negative result showing that, generally speaking, *linear* streaming algorithms cannot be adversarially robust.[5] This result does not rule out non-linear algorithms. Indeed, strong positive results were shown by [6, 18, 26] who constructed (non-linear) adversarially robust algorithms for many problems of interest, with small overhead compared to the oblivious setting. This includes problems such as estimating frequency moments, counting the number of distinct elements in the stream, identifying heavy-hitters in the stream, estimating the median of the stream, entropy estimation, and more. The strong positive results of [6, 18, 26] raise the possibility that adversarial robustness can come "for free" in terms of the additional costs to memory, compared to what is needed in the oblivious setting.

Question 1.1 *Does adversarial streaming require more space than oblivious streaming?*

We provide a positive answer to this question. Specifically, we present a streaming problem for which every adversarially-robust streaming algorithm must use polynomial space, while there exists an oblivious streaming algorithm that uses only polylogarithmic space.

1.1 Streaming against adaptive adversaries

Before describing our new results, we define our setting more precisely. A stream of length m over a domain X consists of a sequence of updates $x_1, \ldots, x_m \in X$. For $i \in [m]$ we write $\vec{x}_i = (x_1, \ldots, x_i)$ to denote the first i updates of the stream. Let $g : X^* \to \mathbb{R}$ be a function (for example, g might count the number of distinct

[5] A streaming algorithm is *linear* if for some (possibly randomized) matrix A, its output depends only on A and Af, where f is the *frequency vector* of the stream.

elements in the stream). At every time step i, after obtaining the next element in the stream x_i, our goal is to output an approximation for $g(\vec{x}_i)$. Throughout the paper we use α for the approximation parameter and β for the confidence parameter.

The adversarial streaming model, in various forms, was considered by [21, 13, 14, 1, 2, 16, 7, 6, 18, 26]. We give here the formulation presented by Ben-Eliezer et al. [6]. The adversarial setting is modeled by a two-player game between a (randomized) `StreamingAlgorithm` and an `Adversary`. At the beginning, we fix a function g. Then the game proceeds in rounds, where in the ith round:

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

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

1.2 Our results

Loosely speaking, we show a reduction from a problem in learning theory, called *adaptive data analysis (ADA)*, to the problem of adversarial streaming. Our results then follow from known impossibility results for the adaptive data analysis problem. In the ADA problem, given a sample S containing n independent samples from some unknown distribution $\mathcal{D}$ over a domain X, the goal is to provide answers to a sequence of adaptively chosen queries w.r.t. $\mathcal{D}$. Importantly, the answers must be accurate w.r.t. the (unknown) underlying distribution $\mathcal{D}$; not just w.r.t. the empirical sample S. In more detail, in the ADA problem, on every time step i we get a query $q_i : X \to \{0,1\}$, and we need to respond with an answer a_i that approximates $q_i(\mathcal{D}) \triangleq \mathbb{E}_{x\sim\mathcal{D}}[q_i(x)]$. Observe that if all of the queries were fixed before the sample S is drawn, then we could simply answer each query q_i with its empirical average $q_i(S) \triangleq \frac{1}{n}\sum_{x\in S} q_i(x)$. Indeed, by the Hoeffding bound, in such a case these answers provide good approximations to the true answers $q_i(\mathcal{D})$. Furthermore, the number of queries ℓ that we can support can be exponential in the sample size n. However, this argument breaks completely when the queries are chosen adaptively based on previous answers given by the mechanism, and the problem becomes much more complex. While, information-theoretically, it is still possible to answer an exponential number of queries (see [11, 5]), it is known that every *computationally efficient* mechanism cannot answer more than n^2 adaptive queries using a sample of size n.

We show that the ADA problem can be phrased as a streaming problem, where the first n elements in the stream are interpreted as "data points" and later

elements in the stream are interpreted as "queries". In order to apply existing impossibility results for the ADA problem, we must overcome the following two main challenges.

Challenge 1 and its resolution. The difficulty in the ADA problem is to maintain accuracy w.r.t. the unknown underlying distribution (and not just w.r.t. the given input sample, which is easy). In the streaming setting, however, there is no underlying distribution, and we cannot require a streaming algorithm to be accurate w.r.t. such a distribution. Instead, we require the streaming algorithm to give accurate answers only w.r.t. the input sample (i.e., w.r.t. the dataset defined by the first n elements in the stream). We then show that if these n elements are sampled i.i.d. from some underlying distribution, then we can use *compression arguments* to show that if the streaming algorithm has small space complexity, and if its answers are accurate w.r.t. the empirical sample, then its answers must in fact be accurate also w.r.t. this underlying distribution. In other words, even though we only require the streaming algorithm to give accurate answers w.r.t. the empirical sample, we show that if it uses small space complexity then its answers must *generalize* to the underlying distribution. This allows us to formulate a link to the ADA problem. We remark that, in the actual construction, we need to introduce several technical modifications in order to make sure that the resulting streaming problem can be solved with small space complexity in the oblivious setting.

Challenge 2 and its resolution. The impossibility results we mentioned for the ADA problem only hold for *computationally efficient* mechanisms.[6] In contrast, we aim for an information-theoretic separation. We therefore cannot apply existing negative results for the ADA problem to our setting as is. Informally, the reason that the negative results for the ADA problem only hold for computationally efficient mechanisms is that their constructions rely on the existence of an efficient *encryption scheme* whose security holds under computational assumptions. We replace this encryption scheme with a different scheme with information-theoretic security against adversaries with *bounded storage* capabilities. Indeed, in our setting, the "adversary" for this encryption scheme will be the streaming algorithm, whose storage capabilities are bounded.

We obtain the following theorem.

Theorem 1.2 *For every w, there exists a streaming problem over domain of size* $\mathrm{poly}(w)$ *and stream length $O(w^5)$ that requires at least w space to be solved in the adversarial setting to within (small enough) constant accuracy, but can be solved in the oblivious setting using space $O(\log^2(w))$.*

1.2.1 Optimality of our results in terms of the flip-number

[6] While there exist information theoretic impossibility results for the ADA problem, they are too weak to give a meaningful result in our context.

The previous works of [6, 18, 26] stated their positive results in terms of the following definition.

Definition 1.3 (Flip number [6]) *Let g be a function defining a streaming problem. The* (α, m)-flip number *of g, denoted as λ, is the maximal number of times that the value of g can change (increase or decrease) by a factor of at least $(1+\alpha)$ during a stream of length m.*

The works of [6, 26, 18] presented general frameworks for transforming an oblivious streaming algorithm $\mathcal{A}$ into an adversarially robust streaming algorithm $\mathcal{B}$ with space complexity (roughly) $\sqrt{\lambda} \cdot \text{Space}(\mathcal{A})$. That is, the results of [6, 26, 18] showed that, generally, adversarial robustness requires space blowup at most (roughly) $\sqrt{\lambda}$ compared to the oblivious setting. For the streaming problem we present (see Theorem 1.2) it holds that the flip-number is $O(w^2)$. That is, for every w, we present a streaming problem with flip-number $\lambda = O(w^2)$, that requires at least $w = \Omega(\sqrt{\lambda})$ space to be solved in the adversarial setting to within (small enough) constant accuracy, but can be solved in the oblivious setting using space $O(\log^2(w))$. This means that, in terms of the dependency of the space complexity in the flip-number, our results are nearly tight. In particular, in terms of λ, our results show that a blowup of $\tilde{\Omega}(\sqrt{\lambda})$ to the space complexity is generally unavoidable in the adversarial setting.

1.2.2 A reduction from adaptive data analysis

Informally, we consider the following streaming problem, which we call the Streaming Adaptive Data Analysis (SADA) problem. On every time step $i \in [m]$ we get an update $x_i \in X$. We interpret the first n updates in the stream $x_1, \ldots, x_n$ as "data points", defining a multiset $S = \{x_1, \ldots, x_n\}$. This multiset does not change after time n.

The next updates in the stream (starting from time $i = n+1$) define "queries" $q : X \to \{0, 1\}$ that should be evaluated by the streaming algorithm on the multiset S. That is, for every such query q, the streaming algorithm should respond with an approximation of $q(S) = \frac{1}{n}\sum_{x \in S} q(x)$. A technical issue here is that every such query is described using $|X|$ bits (represented using its truth table), and hence, cannot be specified using a single update in the stream (which only consists of $\log |X|$ bits). Therefore, every query is specified using $|X|$ updates in the stream. Specifically, starting from time $i = n+1$, every bulk of $|X|$ updates defines a query $q : X \to \{0, 1\}$. At the end of every such bulk, the goal of the streaming algorithm is to output (an approximation for) the average of q on the multiset S. On other time steps, the streaming algorithm should output 0.

As we mentioned, we use *compression arguments* to show that if the streaming algorithm is capable of accurately approximating the average of every such query on the multiset S, and if it uses small space, then when the "data points" (i.e., the elements in the first n updates) are sampled i.i.d. from some distribution $\mathcal{D}$ on X, then the answers given by the streaming algorithm must in fact be accurate also w.r.t. the expectation of these queries on $\mathcal{D}$. This means that the

existence of a streaming algorithm for the SADA problem implies the existence of an algorithm for the adaptive data analysis (ADA) problem, with related parameters. Applying known impossibility results for the ADA problem, this results in a contradiction. However, as we mentioned, the impossibility results we need for the ADA problem only hold for *computationally efficient* mechanisms. Therefore, the construction outlined here only rules out *computationally efficient* adversarially-robust streaming algorithms for the SADA problem. To get an information-theoretic separation, we modify the definition of the SADA problem and rely on cryptographic techniques from the *bounded storage* model.

Remark 1.4 *In Section 6 we outline a variant of the SADA problem, which is more "natural" in the sense that the function to estimate is symmetric. That is, it does not depend on the order of elements in the stream. For this variant, we show a computational separation (assuming the existence of a sub-exponentially secure private-key encryption scheme).*

2 Preliminaries

Our results rely on tools and techniques from learning theory (in particular *adaptive data analysis* and *compression arguments*), and cryptography (in particular *pseudorandom generators* and *encryption schemes*). We now introduce the needed preliminaries.

2.1 Adaptive data analysis

A *statistical query* over a domain X is specified by a predicate $q : X \to \{0,1\}$. The value of a query q on a distribution $\mathcal{D}$ over X is $q(\mathcal{D}) = \mathbb{E}_{x\sim\mathcal{D}}[q(x)]$. Given a database $S \in X^n$ and a query q, we denote the empirical average of q on S as $q(S) = \frac{1}{n}\sum_{x\in S} q(x)$.

In the adaptive data analysis (ADA) problem, the goal is to design a *mechanism* $\mathcal{M}$ that answers queries w.r.t. an unknown distribution $\mathcal{D}$ using only i.i.d. samples from it. Our focus is the case where the queries are chosen adaptively and adversarially. Specifically, $\mathcal{M}$ is a stateful algorithm that holds a collection of samples $(x_1, \ldots, x_n)$, takes a statistical query q as input, and returns an answer z. We require that when $x_1, \ldots, x_n$ are independent samples from $\mathcal{D}$, then the answer z is close to $q(\mathcal{D})$. Moreover we require that this condition holds for every query in an adaptively chosen sequence $q_1, \ldots, q_\ell$. Formally, we define an accuracy game $\mathsf{Acc}_{n,\ell,\mathcal{M},\mathbb{A}}$ between a mechanism $\mathcal{M}$ and a stateful *adversary* $\mathbb{A}$ (see Algorithm 1).

Definition 2.1 ([11]) *A mechanism $\mathcal{M}$ is (α,β)-*statistically-accurate for ℓ adaptively chosen statistical queries given n samples *if for every adversary $\mathbb{A}$ and every distribution $\mathcal{D}$,*

$$\Pr_{\substack{S\sim\mathcal{D}^n \\ \mathsf{Acc}_{n,\ell,\mathcal{M},\mathbb{A}}(S)}}\left[\max_{i\in[\ell]} |q_i(\mathcal{D}) - z_i| \leq \alpha\right] \geq 1-\beta. \tag{1}$$

Algorithm 1 The Accuracy Game $\mathsf{Acc}_{n,\ell,\mathcal{M},\mathbb{A}}$.

Input: A database $S \in X^n$.

1. The database S is given to $\mathcal{M}$.
2. For $i = 1$ to ℓ,
   (a) The adversary $\mathbb{A}$ chooses a statistical query q_i.
   (b) The mechanism $\mathcal{M}$ gets q_i and outputs an answer z_i.
   (c) The adversary $\mathbb{A}$ gets z_i.
3. Output the transcript $(q_1, z_1, \ldots, q_\ell, z_\ell)$.

Remark 2.2 *Without loss of generality, in order to show that a mechanism $\mathcal{M}$ is (α, β)-statistically-accurate (as per Definition 2.1), it suffices to consider only* deterministic *adversaries $\mathbb{A}$. Indeed, given a randomized adversary $\mathbb{A}$, if requirement (1) holds for every fixture of its random coins, then it also holds when the coins are random.*

We use a similar definition for empirical accuracy:

Definition 2.3 ([11]) *A mechanism $\mathcal{M}$ is* (α, β)-empirically accurate for ℓ adaptively chosen statistical queries given a database of size n *if for every adversary $\mathbb{A}$ and every database S of size n,*

$$\Pr_{\mathsf{Acc}_{n,\ell,\mathcal{M},\mathbb{A}}(S)}\left[\max_{i\in[\ell]} |q_i(S) - z_i| \leq \alpha\right] \geq 1 - \beta.$$

2.2 Transcript compressibility

An important notion that allows us to argue about the utility guarantees of an algorithm that answers adaptively chosen queries is *transcript compressibility*, defined as follows.

Definition 2.4 ([10]) *A mechanism $\mathcal{M}$ is* transcript compressible *to $b(n, \ell)$ bits if for every deterministic adversary $\mathbb{A}$ there is a set of transcripts $H_{\mathbb{A}}$ of size $|H_{\mathbb{A}}| \leq 2^{b(n,\ell)}$ such that for every dataset $S \in X^n$ we have*

$$\Pr\left[\mathsf{Acc}_{n,\ell,\mathcal{M},\mathbb{A}}(S) \in H_{\mathbb{A}}\right] = 1.$$

The following theorem shows that, with high probability, for every query generated throughout the interaction with a transcript compressible mechanism it holds that its empirical average is close to its expectation.

Theorem 2.5 ([10]) *Let $\mathcal{M}$ be transcript compressible to $b(n, \ell)$ bits, and let $\beta > 0$. Then, for every adversary $\mathbb{A}$ and for every distribution $\mathcal{D}$ it holds that*

$$\Pr_{\substack{S\sim\mathcal{D}^n \\ \mathsf{Acc}_{n,\ell,\mathcal{M},\mathbb{A}}(S)}} \left[\exists i \text{ such that } |q_i(S) - q_i(\mathcal{D})| > \alpha\right] \leq \beta,$$

where

$$\alpha = O\left(\sqrt{\frac{b(n,\ell) + \ln(\ell/\beta)}{n}}\right).$$

2.3 Pseudorandom generators in the bounded storage model

Our results rely on the existence of pseudorandom generators providing information theoretic security against adversaries with bounded storage capabilities. This security requirement is called the *bounded storage model*. This model was introduced by Maurer [20], and has generated many interesting results, e.g., [20, 8, 4, 3, 9, 12, 19, 17]. We give here the formulation presented by Vadhan [24].

The bounded storage model utilizes a short seed $K \in \{0,1\}^b$ (unknown to the adversary) and a long stream of public random bits $X_1, X_2, \ldots$ (known to all parties). A *bounded storage model (BSM) pseudorandom generator* is a function $\mathrm{PRG} : \{0,1\}^a \times \{0,1\}^b \to \{0,1\}^c$, typically with $b, c \ll a$. Such a scheme is to be used as follows. Initially, two (honest) parties share a seed $K \in \{0,1\}^b$ (unknown to the adversary). At time $t \in [T]$, the next a bits of the public stream $(X_{(t-1)a}, \ldots, X_{ta})$ are broadcast. The adversary is allowed to listen to this stream, however, it cannot store all of it as it has bounded storage capabilities. The honest parties apply $\mathrm{PRG}(\cdot, K)$ to this stream obtain c pseudorandom bits, denoted as $Y_t \in \{0,1\}^c$.

We now formally define security for a BSM pseudorandom generator. Let ρa be the bound on the storage of the adversary $\mathcal{A}$ (we refer to ρ as the *storage rate* of the adversary). We write $S_t \in \{0,1\}^{\rho a}$ to denote the state of the adversary at time t. We consider the adversary's ability to distinguish two experiments — the "real" one, in which the pseudorandom generator is used, and an "ideal" one, in which truly random bits are used. Let $\mathcal{A}$ be an arbitrary function representing the way the adversary updates its storage and attempts to distinguish the two experiments at the end.

Real Experiment:

- Let $X = (X_1, X_2, \ldots, X_{Ta})$ be a sequence of uniformly random bits, let $K \leftarrow \{0,1\}^b$ be the key, and let the adversary's initial state by $S_0 = 0^{\rho a}$.
- For $t = 1, \ldots, T$:
  - Let $Y_t = \mathrm{PRG}\left(X_{(t-1)a+1}, \ldots, X_{ta}, K\right) \in \{0,1\}^c$ be the pseudorandom bits.
  - Let $S_t = \mathcal{A}\left(Y_1, \ldots, Y_{t-1}, S_{t-1}, X_{(t-1)a+1}, \ldots, X_{ta}\right) \in \{0,1\}^{\rho a}$ be the adversary's new state.
- Output $\mathcal{A}\left(Y_1, \ldots, Y_T, S_T, K\right)$

Ideal Experiment:

- Let $X = (X_1, X_2, \ldots, X_{Ta})$ be a sequence of uniformly random bits, let $K \leftarrow \{0,1\}^b$ be the key, and let the adversary's initial state by $S_0 = 0^{\rho a}$.

- For $t = 1, \ldots, T$:
  - Let $Y_t \leftarrow \{0,1\}^c$ be truly random bits.
  - Let $S_t = \mathcal{A}\left(Y_1, \ldots, Y_{t-1}, S_{t-1}, X_{(t-1)a+1}, \ldots, X_{ta}\right) \in \{0,1\}^{\rho a}$ be the adversary's new state.
- Output $\mathcal{A}\left(Y_1, \ldots, Y_T, S_T, K\right) \in \{0,1\}$.

Note that at each time step we give the adversary access to all the past Y_i's "for free" (i.e. with no cost in the storage bound), and in the last time step, we give the adversary the adversary the key K.

Definition 2.6 ([24]) *We call* $\mathrm{PRG} : \{0,1\}^a \times \{0,1\}^b \to \{0,1\}^c$ *an* ε-secure BSM pseudorandom generator *for storage rate* ρ *if for every adversary* $\mathcal{A}$ *with storage bound* ρa*, and every* $T \in \mathbb{N}$*, the adversary* $\mathcal{A}$ *distinguishes between the real and ideal experiments with advantage at most* $T\varepsilon$*. That is,*

$$\left| \Pr_{\text{real}} \left[\mathcal{A}\left(Y_1, \ldots, Y_T, S_T, K\right) = 1\right] - \Pr_{\text{ideal}} \left[\mathcal{A}\left(Y_1, \ldots, Y_T, S_T, K\right) = 1\right] \right| \leq T \cdot \varepsilon$$

Remark 2.7 *No constraint is put on the computational power of the adversary except for the storage bound of* ρa *(as captured by* $S_t \in \{0,1\}^{\rho a}$*). This means that the distributions of* $(Y_1, \ldots, Y_T, S_T, K)$ *in the real and ideal experiments are actually close in a statistical sense – they must have statistical difference at most* $T \cdot \varepsilon$*.*

We will use the following result of Vadhan [24]. We remark that this is only a special case of the results of Vadhan, and refer the reader to [24] for a more detailed account.

Theorem 2.8 ([24]) *For every* $a \in \mathbb{N}$*, every* $\varepsilon > \exp\left(-a/2^{O(\log^* a)}\right)$*, and every* $c \leq a/4$*, there is a BSM pseudorandom generator* $\mathrm{PRG} : \{0,1\}^a \times \{0,1\}^b \to \{0,1\}^c$ *such that*

1. PRG *is* ε*-secure for storage rate* $\rho \leq 1/2$*.*
2. PRG *has key length* $b = O(\log(a/\varepsilon))$*.*
3. *For every key* K*,* $\mathrm{PRG}(\cdot, K)$ *reads at most* $h = O(c + \log(1/\varepsilon))$ *bits from the public stream (nonadaptively).*
4. PRG *is computable in time* $\mathrm{poly}(h, b)$ *and uses workspace* $\mathrm{poly}(\log h, \log b)$ *in addition to the* h *bits read from the public stream and the key of length* b*.*

3 The Streaming Adaptive Data Analysis (SADA) Problem

In this section we introduce a streaming problem, which we call the Streaming Adaptive Data Analysis (SADA) problem, for which we show a strong positive result in the oblivious setting and a strong negative result in the adversarial setting.

Let $X = \{0,1\}^d \times \{0,1\}^b$ be a data domain, let $\gamma \geq 0$ be a fixed constant, and let $\mathrm{PRG} : \{0,1\}^a \times \{0,1\}^b \to \{0,1\}^c$ be a BSM pseudorandom generator, where $c = 1$. We consider the following streaming problem. On every time step $i \in [m]$ we get an update $x_i = (p_i, k_i) \in X$. We interpret the first n updates in the stream $x_1, \ldots, x_n$ as pairs of "data points" and their corresponding "keys". Formally, we denote by S the multiset containing the pairs $x_1, \ldots, x_n$. For technical reasons,[7] the multiset S also contains $\frac{\gamma n}{1-\gamma}$ copies of some arbitrary element $\perp$. This multiset does not change after time n.

Starting from time $j = n + 1$, each bulk of $(a + 1) \cdot 2^d$ updates re-defines a "function" (or a "query") that should be evaluated by the streaming algorithm on the multiset S. This function is defined as follows.

1. For $p \in \{0,1\}^d$ (in lexicographic order) do
   (a) Let $x^{p,1}, \ldots, x^{p,a} \in X$ denote the next a updates, and let $\Gamma^p \in \{0,1\}^a$ be the bitstring containing the first bit of every such update.
   (b) Let $x^{p,a+1}$ denote the next update, and let σ^p denote its first bit.
   (c) For every $k \in \{0,1\}^b$, let $Y_k^p = \mathrm{PRG}(\Gamma^p, k)$ and define $f(p,k) = \sigma^p \oplus Y_k^p$.
2. Also set $f(\perp) = 1$.

This defines a function $f : \left(\{0,1\}^d \times \{0,1\}^b\right) \cup \{\perp\} \to \{0,1\}$.

Definition 3.1 (The (a, b, d, m, n, γ)-SADA Problem) *At the end of every such bulk, defining a function f, the goal of the streaming algorithm is to output (an approximation for) the average of f on the multiset S. On other time steps, the streaming algorithm should output 0.*

Remark 3.2 *In the definition above, m is the total number of updates (i.e., the length of the stream), n is the number of updates that we consider as "date points", γ is a small constant, and a, b, d are the parameters defining the domain and the* PRG.

4 An Oblivious Algorithm for the SADA Problem

In the oblivious setting, we can easily construct a streaming algorithm for the SADA problem using *sampling*. Specifically, throughout the first phase of the execution (during the first n time steps) we maintain a small representative sample from the "data items" (and their corresponding "keys") from the stream. In the second phase of the execution we use this sample in order to answer the given queries. Consider Algorithm `ObliviousSADA`, specified in Algorithm 2. We now analyze its utility guarantees. We will assume that `ObliviousSADA` is executed with a sampling algorithm SAMP that returns a uniformly random sample. This can be achieved, e.g., using Reservoir Sampling [25].

[7] Specifically, recall that the error in the ADA problem is additive while the error in the streaming setting is multiplicative. We add a (relatively small) number of $\perp$'s to S in order to bridge this technical gap.

Algorithm 2 `ObliviousSADA`

Setting: On every time step we obtain the next update, which is an element of $X = \{0,1\}^d \times \{0,1\}^b$.

Algorithm used: A sampling algorithm SAMP that operates on a stream of elements from the domain X and maintains a representative sample.

1. Instantiate algorithm SAMP.
2. REPEAT n times
   (a) Obtain the next update in the stream $x = (p, k)$.
   (b) Output 0.
   (c) Feed the update x to SAMP.
3. Feed (one by one) $\frac{\gamma n}{1-\gamma}$ copies of $\perp$ to SAMP.
4. Let D denote the sample produced by algorithm SAMP.
5. REPEAT (each iteration of this loop spans over $2^d(a+1)$ updates that define a query)
   (a) Let v denote the multiplicity of $\perp$ in D, and set $F = \frac{v}{|D|}$.
   (b) For every $p \in \{0,1\}^d$ in lexicographic order do
      i. Denote $K_p = \{k : (p,k) \in D\}$. That is, K_p is the set of all keys k such that (p,k) appears in the sample D.
      ii. REPEAT a times
         – Obtain the next update x
         – For every $k \in K_p$, feed the first bit of x to $\mathrm{PRG}(\cdot, k)$.
         – Output 0.
      iii. For every $k \in K_p$, obtain a bit Y_k from $\mathrm{PRG}(\cdot, k)$.
      iv. Obtain the next update and let σ be its first bit (and output 0).
      v. For every $k \in K_p$ such that $\sigma \oplus Y_k = 1$: Let $v_{(p,k)}$ denote the multiplicity of (p,k) in D, and set $F \leftarrow F + \frac{v_{(p,k)}}{|D|}$.
   (c) Output F.

Theorem 4.1 *Algorithm* `ObliviousSADA` *is (α, β)-accurate for the SADA problem in the oblivious setting.*

Proof. Fix the stream $\vec{x}_m = (x_1, \ldots, x_m)$. We assume that `ObliviousSADA` is executed with a sampling algorithm SAMP that returns a sample D containing $|D|$ elements, sampled uniformly and independently from $S = (x_1, \ldots, x_n, \perp, \ldots, \perp)$. This can be achieved, e.g., using Reservoir Sampling [25]. As the stream is fixed (and it is of length m), there are at most m different queries that are specified throughout the execution. By the Chernoff bound, assuming that $|D| \geq \Omega\left(\frac{1}{\alpha^2\gamma}\ln(\frac{m}{\beta})\right)$, with probability at least $1-\beta$, for every query f throughout the execution we have that $f(D) \in (1 \pm \alpha) \cdot f(S)$. The theorem now follows by observing that the answers given by algorithm `ObliviousSADA` are exactly the empirical average of the corresponding queries on D. □

Observation 4.2 *For constant α, β, γ, using the pseudorandom generator from Theorem 2.8, algorithm* `ObliviousSADA` *uses space $O\left(\left(\log(\frac{1}{\varepsilon}) + b + d\right) \cdot \log(m)\right)$.*

Algorithm 3 `AnswerQueries`

Input: A database $P \in (\{0,1\}^d)^n$ containing n elements from $\{0,1\}^d$.
Setting: On every time step we get a query $q : \{0,1\}^d \to \{0,1\}$.
Algorithm used: An adversarially robust streaming algorithm $\mathcal{A}$ for the SADA problem with (α, β)-accuracy for streams of length m. We abstract the coin tosses of $\mathcal{A}$ using *two* random strings, r_1 and r_2, of possibly unbounded length. Initially, we execute $\mathcal{A}$ with access to r_1, meaning that every time it tosses a coin it gets the next bit in r_1. At some point, we switch the random string to r_2, and henceforth $\mathcal{A}$ gets its coin tosses from r_2.
Algorithm used: BSM pseudorandom generator $\mathrm{PRG} : \{0,1\}^a \times \{0,1\}^b \to \{0,1\}$, as in the definition of the SADA problem.

1. For every $p \in \{0,1\}^d$ sample $k_p \in \{0,1\}^b$ uniformly.
2. Sample $r_1 \in \{0,1\}^\nu$ uniformly, and instantiate algorithm $\mathcal{A}$ with read-once access to bits of r_1. Here ν bounds the number of coin flips made by $\mathcal{A}$.
3. For every $p \in P$, feed the update (p, k_p) to $\mathcal{A}$.
4. Sample $r_2 \in \{0,1\}^\nu$ uniformly, and switch the read-once access of $\mathcal{A}$ to r_2. (The switch from r_1 to r_2 is done for convenience, so that after Step 3 we do not need to "remember" the position for the next coin from r_1.)
5. REPEAT $\ell \triangleq \frac{m-n}{(a+1)\cdot 2^d}$ times
   (a) Obtain the next query $q : \{0,1\}^d \to \{0,1\}$.
   (b) For every $p \in \{0,1\}^d$ do
      i. Sample $\Gamma \in \{0,1\}^a$ uniformly.
      ii. Feed a updates (one by one) to $\mathcal{A}$ s.t. the concatenation of their first bits is Γ.
      iii. Let $Y = \mathrm{PRG}(\Gamma, k_p)$.
      iv. Feed to $\mathcal{A}$ an update whose first bit is $Y \oplus q(p)$.
   (c) Obtain an answer z from $\mathcal{A}$.
   (d) Output z.

Proof. The algorithm maintains a sample D containing $O(\log m)$ elements, where each element is represented using $b + d$ bits. In addition, the pseudorandom generator uses $O(\log(\frac{1}{\varepsilon}))$ bits of memory, and the algorithm instantiates at most $|D| = O(\log m)$ copies of it. □

5 An Impossibility Result for Adaptive Streaming

Suppose that there is an adversarially robust streaming algorithm $\mathcal{A}$ for the SADA problem. We use $\mathcal{A}$ to construct an algorithm that gets a sample P containing n points in $\{0,1\}^d$, and answers adaptively chosen queries $q : \{0,1\}^d \to \{0,1\}$. Consider Algorithm `AnswerQueries`, specified in Algorithm 3.

By construction, assuming that $\mathcal{A}$ is accurate for the SADA problem, we get that `AnswerQueries` is empirically-accurate (w.r.t. its input database P). Formally,

Claim 5.1 *If $\mathcal{A}$ is (α,β)-accurate for the SADA problem, then* `AnswerQueries` *is $\left(\frac{\alpha}{1-\gamma},\beta\right)$-empirically-accurate for $\frac{m-n}{(a+1)\cdot 2^d}$ adaptively chosen statistical queries given a database of size n. Here γ is a fixed constant (mentioned above).*

Proof sketch. Let q denote the query given at some iteration, and let f denote the corresponding function specified to algorithm $\mathcal{A}$ during this iteration. The claim follows from the fact that, by construction, for every (p,k) we have that $f(p,k)=q(p)$. Specifically, w.h.p., the answers given by $\mathcal{A}$ are α-accurate w.r.t. $P\cup\{\perp,\ldots,\perp\}$, and hence, $\frac{\alpha}{1-\gamma}$-accurate w.r.t. P. □

We now show that algorithm `AnswerQueries` is transcript-compressible. To that end, for every choice of $\vec{\Gamma},\vec{k},r_1,r_2$ for the strings Γ, the keys k, and the random bitstrings r_1,r_2 used throughout the execution, let us denote by $\texttt{AnswerQueries}_{\vec{\Gamma},\vec{k},r_1,r_2}$ algorithm `AnswerQueries` after fixing these elements.

Claim 5.2 *If algorithm $\mathcal{A}$ uses space at most w, then, for every $\vec{\Gamma},\vec{k},r_1,r_2$, we have that algorithm $\texttt{AnswerQueries}_{\vec{\Gamma},\vec{k},r_1,r_2}$ is transcript-compressible to w bits.*

Proof sketch. Assuming that the adversary who generates the queries q is deterministic (which is without loss of generality) we get that the entire transcript is determined by the state of algorithm $\mathcal{A}$ at the end of Step 3. □

Remark 5.3 *The "switch" from r_1 to r_2 is convenient in the proof of Claim 5.2. Otherwise, in order to describe the state of the algorithm after Step 3 we need to specify both the internal state of $\mathcal{A}$ and the position for the next coin from r_1.*

Combining Claims 5.1 (empirical accuracy), and 5.2 (transcript-compression), we get the following lemma.

Lemma 5.4 *Suppose that $\mathcal{A}$ is (α,β)-accurate for the SADA problem for streams of length m using memory w. Then for every $\beta'>0$, algorithm* `AnswerQueries` *is $\left(\frac{\alpha}{1-\gamma}+\alpha',\beta+\beta'\right)$-statistically-accurate for $\ell=\frac{m-n}{(a+1)\cdot 2^d}$ queries, where*

$$\alpha'=O\left(\sqrt{\frac{w+\ln(\frac{\ell}{\beta'})}{n}}\right).$$

Proof. Fix a distribution $\mathcal{D}$ over $\{0,1\}^d$ and fix an adversary $\mathbb{A}$ that generates the queries q_i. Consider the execution of the accuracy game Acc (given in Algorithm 1). By Claim 5.1,

$$\Pr_{\substack{S\sim\mathcal{D}^n\\ \mathsf{Acc}_{n,\ell,\texttt{AnswerQueries},\mathbb{A}}(S)}}\left[\exists i \text{ such that } |q_i(S)-z_i|>\frac{\alpha}{1-\gamma}\right]\leq\beta,$$

where the z_i's denote the answers given by the algorithm. In addition, by Claim 5.2 and Theorem 2.5, for every fixing of $\vec{\Gamma},\vec{k},r_1,r_2$ we have that

$$\Pr_{\substack{S\sim\mathcal{D}^n\\ \mathsf{Acc}_{n,\ell,\texttt{AnswerQueries},\mathbb{A}}(S)}}\left[\exists i \text{ such that } |q_i(S)-q_i(\mathcal{D})|>\alpha'\middle|\vec{\Gamma},\vec{k},r_1,r_2\right]\leq\beta', \quad (2)$$

where

$$\alpha' = O\left(\sqrt{\frac{w + \ln(\ell/\beta')}{n}}\right).$$

Since Inequality (2) holds for *every* fixing of $\vec{\Gamma}, \vec{k}, r_1, r_2$, it also holds when sampling them. Therefore, by the triangle inequality and the union bound,

$$\begin{aligned}
&\Pr_{\substack{S\sim\mathcal{D}^n \\ \mathsf{Acc}_{n,\ell,\mathtt{AnswerQueries},\mathbb{A}}(S)}}\left[\exists i \text{ such that } |z_i - q_i(\mathcal{D})| > \frac{\alpha}{1-\gamma} + \alpha'\right] \\
&\leq \Pr_{\substack{S\sim\mathcal{D}^n \\ \mathsf{Acc}_{n,\ell,\mathtt{AnswerQueries},\mathbb{A}}(S)}}\left[\exists i \text{ such that } |q_i(S) - z_i| > \frac{\alpha}{1-\gamma} \text{ or } |q_i(S) - q_i(\mathcal{D})| > \alpha'\right] \\
&\leq \beta + \beta'.
\end{aligned}$$

□

To obtain a contradiction, we rely on the following impossibility result for the ADA problem. Consider an algorithm $\mathcal{M}$ for the ADA problem that gets an input sample $P = (p_1, \ldots, p_n)$ and answers (adaptively chosen) queries q. The impossibility result we use states that if $\mathcal{M}$ computes the answer to every given query q only as a function of the value of q on points from P (i.e., only as a function of $q(p_1), \ldots, q(p_n)$), then, in general, $\mathcal{M}$ cannot answer more than n^2 adaptively chosen queries. An algorithm $\mathcal{M}$ satisfying this restriction is called a *natural* mechanism. Formally,

Definition 5.5 ([15]) *An algorithm that takes a sample P and answers queries q is* natural *if for every input sample P and every two queries q and q' such that $q(p) = q'(p)$ for all $p \in P$, the answers z and z' that the algorithm gives on queries q and q', respectively, are identical if the algorithm is deterministic and identically distributed if the algorithm is randomized. If the algorithm is stateful, then this condition should hold when the algorithm is in any of its possible states.*

We will use the following negative result of Steinke and Ullman [23] (see also [15, 22]).

Theorem 5.6 ([23]) *There exists a constant $c > 0$ such that there is no natural algorithm that is (c, c)-statistically-accurate for $O(n^2)$ adaptively chosen queries given n samples over a domain of size $\Omega(n)$.*

We have already established (in Lemma 5.4) that algorithm `AnswerQueries` is statistically-accurate for $\ell = \frac{m-n}{(a+1)\cdot 2^d}$ adaptively chosen queries, where ℓ can easily be made bigger then n^2 (by taking m to be big enough). We now want to apply Theorem 5.6 to our setting in order to get a contradiction. However, algorithm `AnswerQueries` is not exactly a natural algorithm (though, as we next explain, it is very close to being natural). The issue is that the answers produced by the streaming algorithm $\mathcal{A}$ can (supposedly) depend on the value

Algorithm 4 `AnswerQueriesOTP`

Input: A database $P \in (\{0,1\}^d)^n$ containing n elements from $\{0,1\}^d$.
Setting: On every time step we get a query $q : \{0,1\}^d \to \{0,1\}$.
Algorithm used: An adversarially robust streaming algorithm $\mathcal{A}$ for the SADA problem with (α, β)-accuracy for streams of length m. We abstract the coin tosses of $\mathcal{A}$ using *two* random strings, r_1 and r_2, of possibly unbounded length. Initially, we execute $\mathcal{A}$ with access to r_1, meaning that every time it tosses a coin it gets the next bit in r_1. At some point, we switch the random string to r_2, and henceforth $\mathcal{A}$ gets its coin tosses from r_2.
Algorithm used: BSM pseudorandom generator $\mathrm{PRG} : \{0,1\}^a \times \{0,1\}^b \to \{0,1\}$, as in the definition of the SADA problem.

1. For every $p \in \{0,1\}^d$ sample $k_p \in \{0,1\}^b$ uniformly.
2. Sample $r_1 \in \{0,1\}^\nu$ uniformly, and instantiate algorithm $\mathcal{A}$ with read-once access to bits of r_1. Here ν bounds the number of coin flips made by $\mathcal{A}$.
3. For every $p \in P$, feed the update (p, k_p) to $\mathcal{A}$.
4. Sample $r_2 \in \{0,1\}^\nu$ uniformly, and switch the read-once access of $\mathcal{A}$ to r_2. (The switch from r_1 to r_2 is done for convenience, so that after Step 3 we do not need to "remember" the position for the next coin from r_1.)
5. REPEAT $\ell \triangleq \frac{m-n}{(a+1)\cdot 2^d}$ times
   (a) Obtain the next query $q : \{0,1\}^d \to \{0,1\}$.
   (b) For every $p \in \{0,1\}^d$ do
      i. Sample $\Gamma \in \{0,1\}^a$ uniformly.
      ii. Feed a updates (one by one) to $\mathcal{A}$ s.t. the concatenation of their first bits is Γ.
      iii. If $p \in P$ then let $Y = \mathrm{PRG}(\Gamma, k_p)$. Otherwise sample $Y \in \{0,1\}$ uniformly.
      iv. Feed to $\mathcal{A}$ an update whose first bit is $Y \oplus q(p)$.
   (c) Obtain an answer z from $\mathcal{A}$.
   (d) Output z.

of the given queries outside of the input sample. Therefore, we now tweak algorithm `AnswerQueries` such that it becomes a natural algorithm. The modified construction is given in Algorithm `AnswerQueriesOTP`, where we marked the modifications in red. Consider Algorithm `AnswerQueriesOTP`, specified in Algorithm 4.

Lemma 5.7 *Algorithm `AnswerQueriesOTP` is natural.*

Proof sketch. This follows from the fact that the value of the given queries outside of the input sample P are completely "hidden" from algorithm $\mathcal{A}$ (namely, by the classic "one-time pad" encryption scheme), and by observing that the answer z given by algorithm `AnswerQueriesOTP` on a query q is determined by the state of algorithm $\mathcal{A}$ at the end of the corresponding iteration of Step 5. □

We now argue that the modification we introduced (from `AnswerQueries` to `AnswerQueriesOTP`) has basically no effect on the execution, and hence, al-

gorithm `AnswerQueriesOTP` is both natural and statistically-accurate. This will lead to a contradiction.

Lemma 5.8 *Suppose that $\mathcal{A}$ has space complexity w. Denote $\ell = \frac{m-n}{(a+1)\cdot 2^d}$. If* PRG *is an ε-secure BSM pseudorandom generator against adversaries with storage $O(w+\ell+b\cdot 2^d)$, then for every input database P and every adversary $\mathbb{A}$, the outcome distributions of $\mathsf{Acc}_{n,\ell,\mathtt{AnswerQueries},\mathbb{A}}(P)$ and $\mathsf{Acc}_{n,\ell,\mathtt{AnswerQueriesOTP},\mathbb{A}}(P)$ are within statistical distance $2^d m\varepsilon$.*

Proof. Recall that the outcome of $\mathsf{Acc}_{n,\ell,\mathtt{AnswerQueries},\mathbb{A}}(P)$ is the transcript of the interaction $(q_1, z_1, \ldots, q_\ell, z_\ell)$, where q_i are the queries given by $\mathbb{A}$, and where z_i are the answers given by `AnswerQueries`. We need to show that the distributions of $(q_1, z_1, \ldots, q_\ell, z_\ell)$ during the executions with `AnswerQueries` and `AnswerQueriesOTP` are close. Without loss of generality, we assume that $\mathbb{A}$ is deterministic (indeed, if the lemma holds for every deterministic $\mathbb{A}$ then it also holds for every randomized $\mathbb{A}$). Hence, the transcript $(q_1, z_1, \ldots, q_\ell, z_\ell)$ is completely determined by the answers given by the mechanism. So we only need to show that $(z_1, \ldots, z_\ell)$ is distributed similarly during the two cases. Note that, as we are aiming for constant accuracy, we may assume that each answer z_i is specified using a constant number of bits (otherwise we can alter algorithm $\mathcal{A}$ to make this true while essentially maintaining its utility guarantees).

Now, for every $g \in \{0, 1, 2, \ldots, 2^d\}$, let $\mathtt{AnswerQueries}_g$ denote an algorithm similar to algorithm `AnswerQueries`, except that in Step 5(b)iii, we set $Y = \mathrm{PRG}(\Gamma, k_p)$ if $p \in P$ or if $p \geq g$, and otherwise we sample $Y \in \{0,1\}$ uniformly. Observe that $\mathtt{AnswerQueries}_0 \equiv \mathtt{AnswerQueries}$ and that $\mathtt{AnswerQueries}_{2^d} \equiv \mathtt{AnswerQueriesOTP}$. We now show that for every g it hods that the statistical distance between $\mathsf{Acc}_{n,\ell,\mathtt{AnswerQueries}_g,\mathbb{A}}(P)$ and $\mathsf{Acc}_{n,\ell,\mathtt{AnswerQueries}_{g+1},\mathbb{A}}(P)$ is at most εm, which proves the lemma (by the triangle inequality).

Fix an index $g \in \{0, 1, \ldots, 2^d - 1\}$. Let Acc^*_g be an algorithm that simulates the interaction between $\mathbb{A}$ and $\mathtt{AnswerQueries}_g$ on the database P, except that during an iteration of Step 5b with $p = g$, algorithm Acc^*_g gets Γ and Y as input, where Γ is sampled uniformly and where Y is either sampled uniformly from $\{0,1\}$ or computed as $Y = \mathrm{PRG}(\Gamma, k)$ for some key k sampled uniformly from $\{0,1\}^b$ (unknown to $\mathtt{AnswerQueries}_g$). These two cases correspond to $\mathsf{Acc}_{n,\ell,\mathtt{AnswerQueries}_{g+1},\mathbb{A}}(P)$ and $\mathsf{Acc}_{n,\ell,\mathtt{AnswerQueries}_g,\mathbb{A}}(P)$, respectively.

Observe that Acc^*_g can be implemented with storage space at most $\hat{W} = O(w+\ell+b\cdot 2^d)$, specifically, for storing the internal state of algorithm $\mathcal{A}$ (which is w bits), storing all previous answers $z_1, z_2, \ldots, z_i$ (which is $O(\ell)$ bits), and storing all the keys k_p for $p \neq g$ (which takes at most $b \cdot 2^d$ bits). Note that, as we assume that $\mathbb{A}$ is deterministic, on every step we can compute the next query from the previously given answers.

Now, when Acc^*_g is given truly random bits Y, then it can be viewed as an adversary acting in the ideal experiment for PRG (see Section 2.3), and when Acc^*_g is given pseudorandom bits then it can be viewed as an adversary acting in the real experiment. By Theorem 2.8, assuming that PRG is ε-secure against adversaries with storage $\hat{W}$, then the distribution on the storage of Acc^*_g in the

two cases is close up to statistical distance εm. The lemma now follows from the fact that the sequence of answers $(z_1, \ldots, z_\ell)$ is included in the storage of Acc^*_g. □

Combining Lemma 5.4 (stating that `AnswerQueries` is statistically-accurate) with Lemma 5.8 (stating that `AnswerQueries` and `AnswerQueriesOTP` are close) we get that `AnswerQueriesOTP` must also be statistically-accurate. Formally,

Lemma 5.9 *Suppose that $\mathcal{A}$ is (α, β)-accurate for the SADA problem for streams of length m using memory w, and suppose that* PRG *is an ε-secure BSM pseudorandom generator against adversaries with storage $O(w+\ell+b\cdot 2^d)$, where $\ell = \frac{m-n}{(a+1)\cdot 2^d}$. Then for every $\beta', \varepsilon > 0$, we have that algorithm `AnswerQueriesOTP` is $\left(\frac{\alpha}{1-\gamma} + \alpha', \beta + \beta' + 2^d m\varepsilon\right)$-statistically-accurate for ℓ queries where*

$$\alpha' = O\left(\sqrt{\frac{w + \ln(\frac{\ell}{\beta'})}{n}}\right).$$

So, Lemmas 5.7 and 5.9 state that algorithm `AnswerQueriesOTP` is both natural and statistically-accurate. To obtain a contradiction to Theorem 5.6, we instantiate Lemma 5.9 with the pseudorandom generator from Theorem 2.8. We obtain the following result.

Theorem 5.10 *For every w, there exists a streaming problem over domain of size* $\mathrm{poly}(w)$ *and stream length $O(w^5)$ that requires at least w space to be solved in the adversarial setting to within (small enough) constant accuracy, but can be solved in the oblivious setting using space $O(\log^2(w))$.*

Proof. To contradict Theorem 5.6, we want the (natural) algorithm `AnswerQueriesOTP` to answer more than n^2 queries over a domain of size $\Omega(n)$. So we set $\ell = \frac{m-n}{(a+1)\cdot 2^d} = \Omega(n^2)$ and $d = O(1) + \log n$. Note that with these settings we have $m = \Theta(n^3 \cdot a)$.

By Lemma 5.9, in order to ensure that `AnswerQueriesOTP`'s answers are accurate (to within some small constant), we set $n = \Theta(w + \log(m))$ (large enough). We assume without loss of generality that $w \geq \log(m)$, as we can always increase the space complexity of $\mathcal{A}$. So $n = \Theta(w)$, and $m = \Theta(w^3 \cdot a)$.

In addition, to apply Lemma 5.9, we need to ensure that the conditions on the security of PRG hold. For a small constant $\tau > 0$, we use the pseudorandom generator from Theorem 2.8 with $\varepsilon = \frac{\tau}{m\cdot 2^d} = O(\frac{1}{mn}) = O(\frac{1}{mw})$. To get security against adversaries with storage $O(w + \ell + b \cdot 2^d) = O(w^2 + bw)$, we need to ensure

$$a = \Omega\left(w^2 + bw\right) \qquad \text{and} \qquad b = \Omega\left(\log\left(\frac{a}{\varepsilon}\right)\right) = \Theta(\log(am)).$$

It suffices to take $a = \Theta(w^2)$ and $b = \Theta(\log(wm)) = \Theta(\log(w))$. Putting everything together, with these parameters, by Lemma 5.9, we get that algorithm

`AnswerQueriesOTP` answers $\ell = \Omega(n^2)$ adaptive queries over domain of size $\Omega(n)$, which contradicts Theorem 5.6. This means that an algorithm with space complexity w cannot solve the (a, b, d, m, n, γ)-SADA problem to within (small enough) constant accuracy, where $a = \Theta(w^2)$, and $b = d = O(\log(w))$, and $m = \Theta(w^5)$, and $n = \Theta(w)$.

In contrast, by Observation 4.2, for constant α, β, γ, the oblivious algorithm `ObliviousSADA` uses space $O(\log^2(w))$ in this settings. □

Remark 5.11 *A natural requirement from a function g, defining a streaming problem, is that the desired outcome does not change significantly from one update to the next (such a function is said to be "insensitive"). In the SADA problem, however, this is not the case. Nevertheless, our separation result can be shown to hold also for such an insensitive function. For example, for a parameter $k \in \mathbb{N}$, we could modify the definition of the SADA problem to ask for the average of the last k given functions, instead of only the last function. This would limit the changes to at most $1/k$. Our separation continues to hold because in the reduction from the ADA problem we could simply ask every query k times.*

6 A Computational Separation

In the previous sections we presented a streaming problem that can be solved in the oblivious setting using small space complexity, but requires large space complexity to be solved in the adversarial setting. Even though this provides a strong separation between adversarial streaming and oblivious streaming, a downside of our result is that the streaming problem we present (the SADA problem) is somewhat unnatural.

Question 6.1 *Is there a "natural" streaming problem for which a similar separation holds?*

In particular, one of the "unnatural" aspects of the SADA problem is that the target function depends on the *order* of the elements in the stream (i.e., it is an asymmetric function). Asymmetric functions can sometimes be considered "natural" in the streaming context (e.g., counting the number of inversions in a stream or finding the longest increasing subsequence). However, the majority of the "classical" streaming problems are defined by symmetric functions (e.g., counting the number of distinct elements in the stream or the number of heavy hitters).

Question 6.2 *Is there a* symmetric *streaming problem that can be solved using polylogarithmic space (in the domain size and the stream length) in the oblivious setting, but requires polynomial space in the adversarial setting?*

In this section we provide a positive answer to this question for *computationally efficient* streaming algorithms. That is, unlike our separation from the previous sections (for the SADA problem) which is information theoretic, the

separation we present in this section (for a symmetric target function) is computational. We consider Question 6.2 (its information theoretic variant) to be an important question for future work.

6.1 The SADA2 Problem

Let $\kappa \in \mathbb{N}$ be a security parameter, let $m \in \mathbb{N}$ denote the length of the stream, and let $d \in \mathbb{N}$ and $\gamma \in (0,1)$ be additional parameters. Let $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ be a semantically secure private-key encryption scheme, with key length κ and ciphertext length $\psi = \mathrm{poly}(\kappa)$ for encrypting a message in $\{0,1\}$. We consider a streaming problem over a domain $X = \{0,1\}^{1+d+\log(m)+\psi}$, where an update $x \in X$ has two possible types (the type is determined by the first bit of x):

Data update: $x = (0, p, k) \in \{0,1\} \times \{0,1\}^d \times \{0,1\}^\kappa$,

Query update: $x = (1, p, j, c) \in \{0,1\} \times \{0,1\}^d \times \{0,1\}^{\log m} \times \{0,1\}^\psi$.

We define a function $g : X^* \to [0,1]$ as follows. Let $\vec{x} = \{x_1, \ldots, x_i\}$ be a sequence of updates. For $p \in \{0,1\}^d$, let $x_{i_1}{=}(0, p, k_{i_1}), \ldots, x_{i_\ell}{=}(0, p, k_{i_\ell})$ denote all the "data updates" in $\vec{x}$ with the point p, and let $k_{i_1}, \ldots, k_{i_\ell}$ denote their corresponding keys (some of which may be identical). Now let $k_p = k_{i_1} \wedge \cdots \wedge k_{i_\ell}$. That is, k_p is the bit-by-bit AND of all of the keys that correspond to "data updates" with the point p. Now let S be the set that contains the pair (p, k_p) for every p such that there exists a "data update" in $\vec{x}$ with the point p. Importantly, S is a *set* rather than a multiset. Similarly to the previous sections, we also add special symbols, $\perp_1, \ldots, \perp_{\gamma 2^d}$, to S. Formally, S is constructed as follows.

1. Initiate $S = \{\perp_1, \ldots, \perp_{\gamma 2^d}\}$.
2. For every $p \in \{0,1\}^d$:
   (a) Let $x_{i_1}{=}(0, p, k_{i_1}), \ldots, x_{i_\ell}{=}(0, p, k_{i_\ell})$ denote all the "data updates" in $\vec{x}$ (i.e., updates beginning with 0) that contain the point p.
   (b) If $\ell > 0$ then let $k_p = k_{i_1} \wedge \cdots \wedge k_{i_\ell}$ and add (p, k_p) to S.

We now define the query q that corresponds to $\vec{x}$. First, $q(\perp_1) = \cdots = q(\perp_{\gamma 2^d}) = 1$. Now, for $p \in \{0,1\}^d$, let

$$j_p^{\max} = \max\left\{j : \exists c \in \{0,1\}^\psi \text{ such that } (1, p, j, c) \in \vec{x}\right\}.$$

That is, $j_p^{\max}$ denotes the maximal index such that $(1, p, j_p^{\max}, c)$ appears in $\vec{x}$ for some $c \in \{0,1\}^\psi$. Furthermore, let $x_{i_1}{=}(1, p, j_p^{\max}, c_{i_1}), \ldots, x_{i_\ell}{=}(1, p, j_p^{\max}, c_{i_\ell})$ denote the "query updates" with p and $j_p^{\max}$. Now let $c_p = c_{i_1} \wedge \cdots \wedge c_{i_\ell}$. That is, c_p is the bit-by-bit AND of all of the ciphertexts that correspond to "query

Algorithm 5 `ObliviousSADA2`

Setting: On every time step we obtain the next update, which is an element of $X = \{0,1\}^{1+d+\log(m)+\psi}$.

1. Let D be a sample (multiset) containing $O(\frac{1}{\alpha^2\gamma^2}\ln(\frac{m}{\beta}))$ i.i.d. elements chosen uniformly from $\{0,1\}^d \cup \{\perp_1, \ldots, \perp_{\gamma 2^d}\}$, and let $D_\perp \leftarrow D \cap \{\perp_1, \ldots, \perp_{\gamma 2^d}\}$, and let $D_X \leftarrow D \setminus D_\perp$.
2. For every $p \in D_X$, let $\texttt{inS}_p \leftarrow 0$, let $k_p \leftarrow \vec{1}$, let $j_p \leftarrow 0$, and let $c_p \leftarrow \vec{1}$.
3. REPEAT
   (a) Obtain the next update in the stream x.
   (b) If the first bit of x is 0 then
      i. Denote $x = (0, p, k)$.
      ii. If $p \in D_X$ then let $\texttt{inS}_p \leftarrow 1$ and let $k_p \leftarrow k_p \wedge k$.
   (c) If the first bit of x is 1 then
      i. Denote $x = (1, p, j, c)$.
      ii. If $p \in D_X$ and $j = j_p$ then set $c_p \leftarrow c_p \wedge c$.
      iii. If $p \in D_X$ and $j > j_p$ then set $c_p \leftarrow c$ and $j_p \leftarrow j$.
   (d) Let $v \leftarrow |\{p \in D_X : \texttt{inS}_p = 1\}| + |D_\perp|$ and let $z \leftarrow \frac{|D_\perp|}{v}$.
   (e) For every $p \in D_X$ such that $\texttt{inS}_p = 1$ set $z \leftarrow z + \frac{\mathrm{Dec}(c_p, k_p)}{v}$.
   (f) Output z.

updates" with p and $j_p^{\max}$. If the point p does not appear in any "query update" then we set $c_p = \vec{1}$ by default. The query $q : \left(\{0,1\}^d \times \{0,1\}^\kappa\right) \to \{0,1\}$ is defined as $q(p,k) = \mathrm{Dec}(c_p, k)$.

Finally, the value of the function g on the stream $\vec{x}$ is defined to be

$$g(\vec{x}) = q(S) = \frac{1}{|S|}\left[\gamma 2^d + \sum_{(p,k_p)\in S} q(p,k_p)\right].$$

That is, $g(\vec{x})$ returns the average of q on S. Observe that g is a symmetric function.

Definition 6.3 (The (d, m, κ, γ)-SADA2 Problem) *At every time step $i \in [m]$, after obtaining the next update $x_i \in X$, the goal is to approximate $g(x_1, \ldots, x_i)$.*

6.2 An Oblivious Algorithm for the SADA2 Problem

In this section we present an oblivious streaming algorithm for the SADA2 problem. The algorithm begins by sampling a multiset D containing a small number of random elements from the domain $\{0,1\}^d \cup \{\perp_1, \ldots, \perp_{\gamma 2^d}\}$. The algorithm then proceeds by maintaining the set S and the query q (which are determined by the input stream; as in the definition of the SADA2 problem) only w.r.t. elements that appear in the sample D. As we next explain, in the oblivious setting, this suffices in order to accurately solve the SADA2 problem. Consider algorithm `ObliviousSADA2`, given in Algorithm 5.

Theorem 6.4 *Assume that* $2^d = \Omega(\frac{1}{\gamma}\ln(\frac{m}{\beta}))$ *and* $|D| \geq \Omega(\frac{1}{\alpha^2\gamma^2}\ln(\frac{m}{\beta}))$. *Then* `ObliviousSADA2` *is* (α, β)*-accurate for the SADA2 problem in the oblivious setting.*

Proof. Fix the stream $\vec{x}_m = (x_1, \ldots, x_m)$. Fix a time step $i \in [m]$, and consider the prefix $\vec{x}_i = (x_1, \ldots, x_i)$. Let $S_i = S_i(\vec{x}_i)$ be the *set* and let $q_i = q_i(\vec{x}_i)$ be the *query* defined by $\vec{x}_i$, as in the definition of the SADA2 problem. Consider the multiset $T = \{(p, k_p) : p \in D_X \text{ and } \texttt{inS}_p = 1\} \cup D_\perp$. Let z_i be the answer returned in Step 3f after precessing the update x_i. Observe that z_i is exactly the average of q_i on the multiset T, that is, $z_i = q_i(T)$.

Recall that $|S_i| \geq \gamma 2^d$, and recall that every element in D is sampled uniformly from $\{0,1\}^d \cup \{\perp_1, \ldots, \perp_{\gamma 2^d}\}$. Therefore, $\mathbb{E}_D[|D \cap S_i|] \geq |D| \cdot \frac{\gamma 2^d}{2^d + \gamma 2^d} = |D| \cdot \frac{\gamma}{1+\gamma}$. By the Chernoff bound, assuming that $2^d = \Omega(\frac{1}{\gamma}\ln(\frac{m}{\beta}))$, then with probability at least $1 - \frac{\beta}{m}$ we have that $|D \cap S_i| \geq \frac{\gamma}{2}|D|$. We proceed with the analysis assuming that this is the case.

Now, for every $t \geq \frac{\gamma}{2}|D|$, when conditioning on $|D \cap S_i| = t$ we have that T is a sample containing t i.i.d. elements from S_i. In that case, again using the Chernoff bound, with probability at least $1 - \frac{\beta}{m}$ we have that $z_i = q_i(T) \in (1 \pm \alpha) \cdot q_i(S_i)$, assuming that $t \geq \Omega(\frac{1}{\alpha^2\gamma}\ln(\frac{m}{\beta}))$. This assumption holds when $|D| \geq \Omega(\frac{1}{\alpha^2\gamma^2}\ln(\frac{m}{\beta}))$.

So, for every fixed i, with probability at least $1 - O(\frac{\beta}{m})$ we have that $z_i \in (1 \pm \alpha) \cdot q_i(S_i)$. By a union bound, this holds for every time step i with probability at least $1 - O(\beta)$. □

Observation 6.5 *For constant* α, β, γ, *algorithm* `ObliviousSADA2` *uses space* $\tilde{O}(\log(m) \cdot \log|X|)$, *in addition to the space required by* Dec.

6.3 A Negative Result for the SADA2 Problem

We now show that the SADA2 problem cannot be solved efficiently in the adversarial setting. To that end, suppose we have an adversarially robust streaming algorithm $\mathcal{A}$ for the SADA2 problem, and consider algorithm `AnswerQueries2` that uses $\mathcal{A}$ in order to solve the ADA problem. Recall that in the SADA2 problem the collection of "data updates" is treated as a *set*, while the input to an algorithm for the ADA problem is a *multiset*. In the following claim we show that `AnswerQueries2` is empirically-accurate w.r.t. its input (when treated as a set).

Claim 6.6 *Let* $P \in \left(\{0,1\}^d\right)^*$ *be an input multiset, let* $\tilde{P}$ *be the* set *containing every point that appears in* P, *and assume that* $|\tilde{P}| = n$. *If* $\mathcal{A}$ *is* (α, β)*-accurate for the SADA2 problem, then* `AnswerQueries2`(P) *is* $\left(\alpha + \frac{\gamma \cdot 2^d}{n}, \beta\right)$*-empirically-accurate for* $\frac{m-n}{2^d}$ *adaptively chosen statistical queries w.r.t. the set* $\tilde{P}$.

Proof sketch. Let q denote the query given at some iteration, and let $q_{\vec{x}}$ and $S_{\vec{x}}$ denote the query and the dataset specified by the updates given to algorithm $\mathcal{A}$.

Algorithm 6 `AnswerQueries2`

Input: A database P containing n elements from $\{0,1\}^d$.
Setting: On every time step we get a query $q : \{0,1\}^d \to \{0,1\}$.
Algorithm used: An adversarially robust streaming algorithm $\mathcal{A}$ for the (d, m, κ, γ)-SADA2 problem with (α, β)-accuracy for streams of length m. We abstract the coin tosses of $\mathcal{A}$ using *two* random strings, r_1 and r_2, of possibly unbounded length. Initially, we execute $\mathcal{A}$ with access to r_1, meaning that every time it tosses a coin it gets the next bit in r_1. At some point, we switch the random string to r_2, and henceforth $\mathcal{A}$ gets its coin tosses from r_2.
Algorithm used: Encryption scheme $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$, as in the definition of the SADA2 problem.

1. For every $p \in \{0,1\}^d$ sample $k_p \leftarrow \mathrm{Gen}(1^\kappa)$ independently.
2. Sample $r_1 \in \{0,1\}^\nu$ uniformly, and instantiate algorithm $\mathcal{A}$ with read-once access to bits of r_1. Here ν bounds the number of coin flips made by $\mathcal{A}$.
3. For every $p \in P$, feed the update $(0, p, k_p)$ to $\mathcal{A}$.
4. Sample $r_2 \in \{0,1\}^\nu$ uniformly, and switch the read-once access of $\mathcal{A}$ to r_2. (The switch from r_1 to r_2 is done for convenience, so that after Step 3 we do not need to "remember" the position for the next coin from r_1.)
5. For $j = 1$ to $\ell \triangleq \frac{m-n}{2^d}$ do
   (a) Obtain the next query $q_j : \{0,1\}^d \to \{0,1\}$.
   (b) For every $p \in \{0,1\}^d$ do
      i. Let $c_p = \mathrm{Enc}(q_j(p), k_p)$.
      ii. Feed the update $(1, p, j, c_p)$ to $\mathcal{A}$.
   (c) Obtain an answer z from $\mathcal{A}$.
   (d) Output z.

The claim follows from the fact that, by construction, for every $p \in \tilde{P}$ we have that $q(p) = q_{\vec{x}}(p, k_p)$. Therefore,

$$q_{\vec{x}}(S_{\vec{x}}) = \frac{1}{|S_{\vec{x}}|}\left[\gamma \cdot 2^d + \sum_{p\in\tilde{P}} q(p)\right] = \frac{1}{n+\gamma 2^d}\left[\gamma \cdot 2^d + \sum_{p\in\tilde{P}} q(p)\right]$$
$$= \frac{\frac{\gamma\cdot 2^d}{n}}{1+\frac{\gamma\cdot 2^d}{n}} + \frac{1}{n+\gamma 2^d}\sum_{p\in\tilde{P}} q(p).$$

Therefore, $q_{\vec{x}}(S_{\vec{x}}) \le \frac{\gamma\cdot 2^d}{n} + q(\tilde{P})$, and also $q_{\vec{x}}(S_{\vec{x}}) \ge \frac{1}{n+\gamma 2^d}\sum_{p\in\tilde{P}} q(p)$ which means that $q(\tilde{P}) \le \frac{n+\gamma 2^d}{n} \cdot q_{\vec{x}}(S_{\vec{x}}) \le q_{\vec{x}}(S_{\vec{x}}) + \frac{\gamma 2^d}{n}$. So, whenever the answers given by $\mathcal{A}$ are α-accurate w.r.t. $q_{\vec{x}}(S_{\vec{x}})$, they are also $\left(\alpha + \frac{\gamma 2^d}{n}\right)$-accurate w.r.t. $\tilde{P}$. □

We now show that algorithm `AnswerQueries2` is transcript-compressible. To that end, for every choice of $\vec{k}, r_1, r_2, \vec{r}_{\mathrm{Enc}}$ for the keys k, the random bitstrings r_1, r_2, and the randomness used by Enc at its different executions, let us denote

by $\texttt{AnswerQueries2}_{\vec{k},r_1,r_2,\vec{r}_{\text{Enc}}}$ algorithm `AnswerQueries2` after fixing these elements.

Claim 6.7 *If algorithm $\mathcal{A}$ uses space at most w, then, for every $\vec{k}, r_1, r_2$, we have that algorithm $\texttt{AnswerQueries2}_{\vec{k},r_1,r_2,\vec{r}_{\text{Enc}}}$ is transcript-compressible to w bits.*

Proof sketch. Assuming that the adversary who generates the queries q is deterministic (which is without loss of generality) we get that the entire transcript is determined by the state of algorithm $\mathcal{A}$ at the end of Step 3. □

Similarly to our arguments from Section 5, since algorithm `AnswerQueries2` is both empirically-accurate and transcript-compressible, we get that it is also statistically-accurate. Since we only argued empirical-accuracy when treating the input multiset as a set, we will only argue for statistical-accuracy w.r.t. the uniform distribution, where we have that the difference between a random set and a random multiset is small. Formally,

Lemma 6.8 *Suppose that $\mathcal{A}$ is (α, β)-accurate for the SADA2 problem for streams of length m using memory w. Then for every $\beta' > 0$, algorithm $\texttt{AnswerQueries2}$ is $\left(\tilde{\alpha}, \tilde{\beta}\right)$-statistically-accurate for $\ell = \frac{m-n}{2^d}$ queries w.r.t. the uniform distribution over $\{0,1\}^d$, where $\tilde{\beta} = O\left(\beta + \beta' + \exp\left(-\frac{n^2}{3 \cdot 2^d}\right)\right)$ and*

$$\tilde{\alpha} = O\left(\alpha + \frac{\gamma \cdot 2^d}{n} + \frac{n}{2^d} + \sqrt{\frac{w + \ln(\frac{\ell}{\beta'})}{n}}\right).$$

Proof sketch. The proof is analogous to the proof of Lemma 5.4, with the following addition. Let P be a multiset containing n i.i.d. uniform samples from $\{0,1\}^d$, and let $\tilde{P}$ be the set containing every element of P. As we are considering the uniform distribution on $\{0,1\}^d$, then by the Chernoff bound, with probability at least $1 - \exp(-\frac{n^2}{3 \cdot 2^d})$, it holds that the set $\tilde{P}$ and and the multiset P differ by at most $\frac{n^2}{2 \cdot 2^d}$ points, i.e., by at most an $\frac{n}{2 \cdot 2^d}$-fraction of the points. In that case, for every query q we have that $|q(P) - q(\tilde{P})| \leq \frac{n}{2 \cdot 2^d}$. □

So algorithm `AnswerQueries2` is statistically-accurate. To obtain a contradiction, we modify the algorithm such that it becomes natural. Consider algorithm `AnswerQueries2Natural`. As before, the modifications are marked in red.

Observation 6.9 *Algorithm $\texttt{AnswerQueries2Natural}$ is natural.*

Proof sketch. This follows from the fact that the value of the given queries outside of the input sample P are ignored, and are replaced with (encryptions of) zero. □

The following lemma follows from the assumed security of the encryption scheme.

Algorithm 7 `AnswerQueries2Natural`

Input: A database P containing n elements from $\{0,1\}^d$.
Setting: On every time step we get a query $q : \{0,1\}^d \to \{0,1\}$.
Algorithm used: An adversarially robust streaming algorithm $\mathcal{A}$ for the (d, m, κ, γ)-SADA2 problem with (α, β)-accuracy for streams of length m. We abstract the coin tosses of $\mathcal{A}$ using *two* random strings, r_1 and r_2, of possibly unbounded length. Initially, we execute $\mathcal{A}$ with access to r_1, meaning that every time it tosses a coin it gets the next bit in r_1. At some point, we switch the random string to r_2, and henceforth $\mathcal{A}$ gets its coin tosses from r_2.
Algorithm used: Encryption scheme $(\text{Gen}, \text{Enc}, \text{Dec})$, as in the definition of the SADA2 problem.

1. For every $p \in \{0,1\}^d$ sample $k_p \leftarrow \text{Gen}(1^\kappa)$ independently.
2. Sample $r_1 \in \{0,1\}^\nu$ uniformly, and instantiate algorithm $\mathcal{A}$ with read-once access to bits of r_1. Here ν bounds the number of coin flips made by $\mathcal{A}$.
3. For every $p \in P$, feed the update $(0, p, k_p)$ to $\mathcal{A}$.
4. Sample $r_2 \in \{0,1\}^\nu$ uniformly, and switch the read-once access of $\mathcal{A}$ to r_2. (The switch from r_1 to r_2 is done for convenience, so that after Step 3 we do not need to "remember" the position for the next coin from r_1.)
5. For $j = 1$ to $\ell \triangleq \frac{m-n}{2^d}$ do
   (a) Obtain the next query $q_j : \{0,1\}^d \to \{0,1\}$.
   (b) For every $p \in \{0,1\}^d$ do
      i. If $p \in P$ then let $c_p = \text{Enc}(q_j(p), k_p)$. Otherwise let $c_p = \text{Enc}(0, k_p)$.
      ii. Feed the update $(1, p, j, c_p)$ to $\mathcal{A}$.
   (c) Obtain an answer z from $\mathcal{A}$.
   (d) Output z.

Lemma 6.10 *Suppose that* $(\text{Gen}, \text{Enc}, \text{Dec})$ *is semantically secure private-key encryption scheme with key length* $\kappa = \kappa(m)$ *against adversaries with time* $\text{poly}(m)$. *Fix* $\alpha \in (0,1)$. *Let* $\mathbb{A}$ *be a data analyst with running time* $\text{poly}(m)$. *For a mechanism* $\mathcal{M}$ *that answers queries, consider the interaction between* $\mathcal{M}$ *and* $\mathbb{A}$ *, and let* E *denote the event that* $\mathcal{M}$ *failed to be* α*-statistically accurate at some point during the interaction. Then, for an input database* P *sampled uniformly from* $\{0,1\}^d$ *it holds that*

$$\left| \Pr_{P,\mathbb{A},\texttt{AnswerQueries2}(P)}[E] - \Pr_{P,\mathbb{A},\texttt{AnswerQueries2Natural}(P)}[E] \right| \leq \text{negl}(\kappa).$$

The proof of Lemma 6.10 is straightforward from the definition of security. We give here the details for completeness. To that end, let us recall the formal definition of security of an encryption scheme. Consider a pair of oracles $\mathcal{E}_0$ and $\mathcal{E}_1$, where $\mathcal{E}_1(k_1, \ldots, k_N, \cdot)$ takes as input an index of a key $i \in [N]$ and a message M and returns $\text{Enc}(M, k_i)$, and where $\mathcal{E}_0(k_1, \ldots, k_N, \cdot)$ takes the same input but returns $\text{Enc}(0, k_i)$. An encryption scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ is *secure* if no computationally efficient adversary can tell whether it is interacting with $\mathcal{E}_0$ or with $\mathcal{E}_1$. Formally,

Algorithm 8 An adversary $\mathcal{B}$ for the encryption scheme

Algorithm used: An adversarially robust streaming algorithm $\mathcal{A}$ for the (d, m, κ, γ)-SADA2 problem with (α, β)-accuracy for streams of length m.
Algorithm used: A data analyst $\mathbb{A}$ that outputs queries and obtains answers.
Algorithm used: Encryption scheme $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$.
Oracle access: $\mathcal{E}_b(k_1, \ldots, k_N, \cdot)$ where $b \in \{0,1\}$ and where $N = 2^d$ and $k_1, \ldots, k_N \leftarrow \mathrm{Gen}(1^\kappa)$.

1. Let P be a multiset containing n uniform samples from $\{0,1\}^d$.
2. For every $p \in P$ sample $\bar{k}_p \leftarrow \mathrm{Gen}(1^\kappa)$ independently.
3. Instantiate algorithm $\mathcal{A}$.
4. For every $p \in P$, feed the update $(0, p, \bar{k}_p)$ to $\mathcal{A}$.
5. Instantiate the data analyst $\mathbb{A}$.
6. For $j = 1$ to $\ell \triangleq \frac{m-n}{2^d}$ do
   (a) Obtain the next query $q_j : \{0,1\}^d \to \{0,1\}$ from the data analyst $\mathbb{A}$.
   (b) For every $p \in \{0,1\}^d$ do
      i. If $p \in P$ then let $c_p = \mathrm{Enc}(q_j(p), \bar{k}_p)$. Otherwise let $c_p \leftarrow \mathcal{E}_b(p, q_j(p))$.
      ii. Feed the update $(1, p, j, c_p)$ to $\mathcal{A}$.
   (c) Obtain an answer z from $\mathcal{A}$, and give z to $\mathbb{A}$.
7. Output 1 if and only if event E occurs.

Definition 6.11 *Let $m : \mathbb{R} \to \mathbb{R}$ be a function. An encryption scheme* $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ *is* m-secure *if for every $N = \mathrm{poly}(m(\kappa))$, and every* $\mathrm{poly}(m(\kappa))$*-time adversary $\mathcal{B}$, the following holds.*

$$\left| \Pr_{\substack{k_1,\ldots,k_N \\ \mathcal{B},\mathrm{Enc}}} \left[\mathcal{B}^{\mathcal{E}_0(k_1,\ldots,k_N,\cdot)} = 1 \right] - \Pr_{\substack{k_1,\ldots,k_N \\ \mathcal{B},\mathrm{Enc}}} \left[\mathcal{B}^{\mathcal{E}_1(k_1,\ldots,k_N,\cdot)} = 1 \right] \right| = \mathrm{negl}(\kappa),$$

where the probabilities are over sampling $k_1, \ldots, k_N \leftarrow \mathrm{Gen}(1^\kappa)$ and over the randomness of $\mathcal{B}$ and Enc.

Remark 6.12 *When m is the identity function we simply say that* $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ *is* secure. *Note that in this case, security holds against all adversaries with runtime polynomial in the security parameter κ. We will further assume the existence of a* sub-exponentially secure *encryption scheme. By that we mean that there exist a constant $\tau > 0$ such that* $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ *is m-secure for $m(\kappa) = 2^{\kappa^\tau}$. That is, we assume the existence of an encryption scheme in which security holds agains all adversaries with runtime polynomial in 2^{κ^τ}.*

To prove Lemma 6.10 we construct an adversary $\mathcal{B}$ for $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ such that its advantage in breaking the security of this scheme is exactly the difference in the probability of event E between the execution with `AnswerQueries2` or with `AnswerQueries2Natural`. This implies that the difference between these two probabilities is negligible.

Proof of Lemma 6.10. Let $\mathbb{A}$ be a data analyst with running time $\text{poly}(m)$, and consider algorithm $\mathcal{B}$. First observe that if $\mathcal{A}$ and $\mathbb{A}$ are computationally efficient (run in time $\text{poly}(m)$) then so is algorithm $\mathcal{B}$.

Now observe that when the oracle is $\mathcal{E}_1$ and when $k_1, \ldots, k_N$ are chosen randomly from $\text{Gen}(1^\kappa)$ then $\mathcal{B}^{\mathcal{E}_1(k_1,\ldots,k_N,\cdot)}$ simulates the interaction between $\mathbb{A}$ and `AnswerQueries2` on a uniformly sampled database P. Similarly, when the oracle is $\mathcal{E}_0$ and when $k_1, \ldots, k_N$ are chosen randomly from $\text{Gen}(1^\kappa)$ then $\mathcal{B}^{\mathcal{E}_0(k_1,\ldots,k_N,\cdot)}$ simulates the interaction between $\mathbb{A}$ and `AnswerQueries2Natural` on a uniformly sampled database P. Thus,

$$\left| \Pr_{P,\mathbb{A},\texttt{AnswerQueries2}(P)}[E] - \Pr_{P,\mathbb{A},\texttt{AnswerQueries2Natural}(P)}[E] \right|$$

$$= \left| \Pr_{\substack{k_1,\ldots,k_N \\ \mathcal{B},\text{Enc}}} \left[\mathcal{B}^{\mathcal{E}_1(k_1,\ldots,k_N,\cdot)} = 1\right] - \Pr_{\substack{k_1,\ldots,k_N \\ \mathcal{B},\text{Enc}}} \left[\mathcal{B}^{\mathcal{E}_0(k_1,\ldots,k_N,\cdot)} = 1\right] \right| = \text{negl}(\kappa).$$

□

So, algorithm `AnswerQueries2Natural` is natural, and when $\mathcal{A}$ and $\mathbb{A}$ are computationally efficient, then the probability that `AnswerQueries2Natural` fails to be statistically-accurate is similar to the probability that `AnswerQueries2` fails, which is small. We therefore get the following lemma.

Lemma 6.13 *Algorithm `AnswerQueries2Natural` is natural. In addition, if* $(\text{Gen}, \text{Enc}, \text{Dec})$ *is an m-secure private-key encryption scheme with key length $\kappa = \kappa(m)$, and if $\mathcal{A}$ is an adversarially robust streaming algorithm for the (d, m, κ, γ)-SADA2 problem with space w and runtime* $\text{poly}(m)$*, then algorithm `AnswerQueries2Natural` is $\left(\tilde{\alpha}, \tilde{\beta}\right)$-statistically-accurate for $\ell = \frac{m-n}{2^d}$ queries w.r.t. the uniform distribution over $\{0,1\}^d$, and w.r.t. a data analyst $\mathbb{A}$ with running time* $\text{poly}(m)$*, where $\tilde{\beta} = O\left(\beta + \beta' + \exp\left(-\frac{n^2}{3 \cdot 2^d}\right) + \text{negl}(\kappa)\right)$ and*

$$\tilde{\alpha} = O\left(\alpha + \frac{\gamma \cdot 2^d}{n} + \frac{n}{2^d} + \sqrt{\frac{w + \ln(\frac{\ell}{\beta'})}{n}}\right).$$

We now restate Theorem 5.6, in which we simplified the results of Steinke and Ullman. In this section we use the stronger formulation of their results, given as follows.

Theorem 6.14 ([23]) *There exists a constant $c > 0$ such that no natural algorithm is (c, c)-statistically-accurate for $O(n^2)$ adaptively chosen queries given n samples over a domain of size $\Omega(n)$. Furthermore, this holds even when assuming that the data analyst is computationally efficient (runs in time* $\text{poly}(n^2)$*) and even when the underlying distribution is the uniform distribution.*

Combining Lemma 6.13 with Theorem 6.14 we obtain the following result.

Theorem 6.15 *Assume the existence of a sub-exponentially secure private-key encryption scheme. Then, the* $(d{=}\Theta(\log m), m, \kappa{=}\operatorname{polylog}(m), \gamma{=}\Theta(1))$*-SADA2 problem can be solved in the oblivious setting to within constant accuracy using space* $\operatorname{polylog}(m)$ *and using* $\operatorname{polylog}(m)$ *runtime (per update). In contrast, every adversarially robust algorithm for this problem with* $\operatorname{poly}(m)$ *runtime per update must use space* $\operatorname{poly}(m)$*.*

References

1. K. J. Ahn, S. Guha, and A. McGregor. Analyzing graph structure via linear measurements. In Y. Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 459–467. SIAM, 2012.
2. K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: sparsification, spanners, and subgraphs. In M. Benedikt, M. Krötzsch, and M. Lenzerini, editors, *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 5–14. ACM, 2012.
3. Y. Aumann, Y. Z. Ding, and M. O. Rabin. Everlasting security in the bounded storage model. *IEEE Transactions on Information Theory*, 48(6):1668–1680, 2002.
4. Y. Aumann and M. O. Rabin. Information theoretically secure communication in the limited storage space model. In *Annual International Cryptology Conference*, pages 65–79. Springer, 1999.
5. R. Bassily, K. Nissim, A. D. Smith, T. Steinke, U. Stemmer, and J. Ullman. Algorithmic stability for adaptive data analysis. In D. Wichs and Y. Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 1046–1059. ACM, 2016.
6. O. Ben-Eliezer, R. Jayaram, D. P. Woodruff, and E. Yogev. A framework for adversarially robust streaming algorithms. *CoRR*, abs/2003.14265, 2020.
7. O. Ben-Eliezer and E. Yogev. The adversarial robustness of sampling. *CoRR*, abs/1906.11327, 2019.
8. C. Cachin and U. Maurer. Unconditional security against memory-bounded adversaries. In *Annual International Cryptology Conference*, pages 292–306. Springer, 1997.
9. Y. Z. Ding and M. O. Rabin. Hyper-encryption and everlasting security. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 1–26. Springer, 2002.
10. C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth. Generalization in adaptive data analysis and holdout reuse. In *Advances in Neural Information Processing Systems (NIPS)*, Montreal, December 2015.
11. C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth. Preserving statistical validity in adaptive data analysis. In *ACM Symposium on the Theory of Computing (STOC)*. ACM, June 2015.
12. S. Dziembowski and U. Maurer. Optimal randomizer efficiency in the bounded-storage model. *Journal of Cryptology*, 17(1):5–26, 2004.
13. A. C. Gilbert, B. Hemenway, A. Rudra, M. J. Strauss, and M. Wootters. Recovering simple signals. In *2012 Information Theory and Applications Workshop*, pages 382–391, 2012.

14. A. C. Gilbert, B. Hemenway, M. J. Strauss, D. P. Woodruff, and M. Wootters. Reusable low-error compressive sampling schemes through privacy. In *2012 IEEE Statistical Signal Processing Workshop (SSP)*, pages 536–539, 2012.
15. M. Hardt and J. Ullman. Preventing false discovery in interactive data analysis is hard. In *FOCS*. IEEE, October 19-21 2014.
16. M. Hardt and D. P. Woodruff. How robust are linear sketches to adaptive inputs? In *STOC*, pages 121–130. ACM, June 1-4 2013.
17. D. Harnik and M. Naor. On everlasting security in the hybrid bounded storage model. In *International Colloquium on Automata, Languages, and Programming*, pages 192–203. Springer, 2006.
18. A. Hassidim, H. Kaplan, Y. Mansour, Y. Matias, and U. Stemmer. Adversarially robust streaming algorithms via differential privacy. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
19. C.-J. Lu. Encryption against storage-bounded adversaries from on-line strong extractors. *Journal of Cryptology*, 17(1):27–42, 2004.
20. U. M. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992.
21. I. Mironov, M. Naor, and G. Segev. Sketching in adversarial environments. *SIAM J. Comput.*, 40(6):1845–1870, 2011.
22. K. Nissim, A. D. Smith, T. Steinke, U. Stemmer, and J. Ullman. The limits of post-selection generalization. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 6402–6411, 2018.
23. T. Steinke and J. Ullman. Interactive fingerprinting codes and the hardness of preventing false discovery. In *COLT*, pages 1588–1628, 2015.
24. S. P. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *Journal of Cryptology*, 17(1):43–77, 2004.
25. J. S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985.
26. D. P. Woodruff and S. Zhou. Tight bounds for adversarially robust streams and sliding windows via difference estimators. *CoRR*, abs/2011.07471, 2020.