The White-Box Adversarial Data Stream Model

Miklós Ajtai
Hungarian Academy of Sciences
Budapest, Hungary
miklos.ajtai@gmail.com

Vladimir Braverman
Google Research
Baltimore, MD, USA
vbraverman@google.com

T.S. Jayram
Lawrence Livermore National Laboratories
Livermore, CA, USA
t.s.jayram@gmail.com

Sandeep Silwal
MIT
Cambridge, MA, USA
silwal@mit.edu

Alec Sun
Carnegie Mellon University
Pittsburgh, PA, USA
alecsun@andrew.cmu.edu

David P. Woodruff
Carnegie Mellon University
Pittsburgh, PA, USA
dwoodruf@andrew.cmu.edu

Samson Zhou
Carnegie Mellon University
Pittsburgh, PA, USA
samsonzhou@gmail.com

ABSTRACT

There has been a flurry of recent literature studying streaming algorithms for which the input stream is chosen adaptively by a black-box adversary who observes the output of the streaming algorithm at each time step. However, these algorithms fail when the adversary has access to the internal state of the algorithm, rather than just the output of the algorithm.

We study streaming algorithms in the *white-box adversarial model*, where the stream is chosen adaptively by an adversary who observes the entire internal state of the algorithm at each time step. We show that nontrivial algorithms are still possible. We first give a randomized algorithm for the L_1-heavy hitters problem that outperforms the optimal deterministic Misra-Gries algorithm on long streams. If the white-box adversary is computationally bounded, we use cryptographic techniques to reduce the memory of our L_1-heavy hitters algorithm even further and to design a number of additional algorithms for graph, string, and linear algebra problems. The existence of such algorithms is surprising, as the streaming algorithm does not even have a secret key in this model, i.e., its state is entirely known to the adversary. One algorithm we design is for estimating the number of distinct elements in a stream with insertions and deletions achieving a multiplicative approximation and sublinear space; such an algorithm is impossible for deterministic algorithms.

We also give a general technique that translates any two-player *deterministic* communication lower bound to a lower bound for *randomized* algorithms robust to a white-box adversary. In particular, our results show that for all $p \geq 0$, there exists a constant $C_p > 1$ such that any C_p-approximation algorithm for F_p moment estimation in insertion-only streams with a white-box adversary requires $\Omega(n)$ space for a universe of size n. Similarly, there is a constant $C > 1$ such that any C-approximation algorithm in an insertion-only stream for matrix rank requires $\Omega(n)$ space with a white-box adversary. These results do not contradict our upper bounds since *they assume the adversary has unbounded computational power*. Our algorithmic results based on cryptography thus show a separation between computationally bounded and unbounded adversaries.

Finally, we prove a lower bound of $\Omega(\log n)$ bits for the fundamental problem of deterministic approximate counting in a stream of 0s and 1s, which holds even if we know how many total stream updates we have seen so far at each point in the stream. Such a lower bound for approximate counting with additional information was previously unknown, and in our context, it shows a separation between multiplayer deterministic maximum communication and the white-box space complexity of a streaming algorithm.

CCS CONCEPTS

• **Theory of computation → Streaming models**; **Streaming, sublinear and near linear time algorithms**; **Lower bounds and information complexity**.

KEYWORDS

data streams, adversarial robustness, counting, cryptography

ACM Reference Format:
Miklós Ajtai, Vladimir Braverman, T.S. Jayram, Sandeep Silwal, Alec Sun, David P. Woodruff, and Samson Zhou. 2022. The White-Box Adversarial Data Stream Model. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3517804.3526228

1 INTRODUCTION

In the streaming model of computation, one wants to compute or approximate a predetermined function on a dataset. The dataset is implicitly defined through a sequence of updates, and the goal is to use total space that is sublinear in the size of the dataset. The streaming model captures key resource requirements of algorithms for many database and network tasks where the size of the data

PODS '22, June 12–17, 2022, Philadelphia, PA, USA.

ACM ISBN 978-1-4503-9260-0/22/06.
https://doi.org/10.1145/3517804.3526228

is significantly larger than the available storage, such as logs for network traffic, IoT sensors, financial markets, commercial transactions, and scientific data, e.g., astronomy or bioinformatics.

In the classical *oblivious* streaming model, there exists a stream S of updates $u_1, \ldots, u_m$ that defines an underlying dataset, such as a frequency vector, a graph, or a set of points in Euclidean space. The sequence of updates may be worst-case, but the dataset is fixed in advance and is oblivious to any algorithmic design choices. Although there are examples of fundamental streaming algorithms that are deterministic, many streaming algorithms crucially utilize randomness to achieve meaningful guarantees in sublinear space. For example, the famous AMS sketch [5] for F_2 estimation initializes a random sign vector Z, maintains $\langle Z, f\rangle$ in the stream, and outputs $\langle Z, f\rangle^2$ which is an unbiased estimator to $\|f\|^2$, where f is the underlying frequency vector defined by the stream. However, the analysis demands that the randomness used to generate the sign vector Z is independent of the frequency vector f and in general, the analysis of many randomized algorithms assumes that the randomness of the algorithm is independent of the input. However, such an assumption may not be reasonable [8, 16, 24, 32, 54, 57]; even if the stream is not adversarially generated, a user may need to repeatedly query and update a database based on the responses to previous queries. For example in typical optimization procedures such as stochastic gradient descent, each time step can update the eventual output by an amount based on a previous query. In recommendation systems, a user may choose to remove some suggestions based on personal preference and then query for a new list of recommendations.

(Black-box) adversarial streaming model. Recently, a large body of research has been devoted to studying the (black-box) adversarial streaming model as a means of modeling adversarial data streams. In the (black-box) adversarial streaming model [4, 7, 11–13, 17, 21, 37, 43, 51, 69], the sequence of updates $u_1, \ldots, u_m$ is chosen adaptively rather than being fixed. In particular, the input is chosen by an adversary who repeatedly queries the streaming algorithm for a fixed property of the underlying dataset at each time $t \in [m]$ and determines the update u_{t+1} only after seeing the output of the algorithm after time t. The streaming algorithm must still be correct at all times. In the black-box adversarial streaming model, [4, 13] show that Bernoulli sampling and reservoir sampling can approximately preserve statistics such as densities of certain subsets of the universe and [17] shows that importance sampling can use independent public randomness to approximately solve problems such as k-means centering, linear regression, and graph sparsification.

However, for other important problems such as F_p moment estimation, matrix rank, or estimating the number of distinct elements in the stream, [7, 11, 12, 21, 37, 69] crucially use the fact that the adversary who chooses the input can only see the output of the algorithm. These algorithms essentially work by arguing that it is possible to have the output of the algorithm change only a small number of times, and so only a small amount of internal randomness is revealed, which allows such algorithms to still be correct. However, these algorithms completely fail if the internal state of the algorithm at each point in time is also revealed to an adversary.

White-box adversarial streaming model. In this paper, we introduce the *white-box* adversarial streaming model, where the sequence of updates $u_1, \ldots, u_m$ is chosen adaptively by an adversary who sees the full internal state of the algorithm at all times, including the parameters and the previous randomness used by the algorithm. More formally, we define the white-box adversarial model as a two-player game between StreamAlg, the streaming algorithm, and Adversary. Prior to the beginning of the game, a query Q is fixed, which asks for a fixed function of some underlying dataset implicitly defined by the stream. The game then proceeds across m rounds, where in the t-th round:

(1) Adversary computes an update $u_t \in [n]$ for the stream, which depends on all previous stream updates, all previous internal states of StreamAlg, and all previous randomness used by StreamAlg (and thus also, all previous outputs of StreamAlg).
(2) StreamAlg uses u_t to update its data structures D_t, acquires a fresh batch R_t of random bits, and outputs a response A_t to the query Q.
(3) Adversary observes the response A_t, the internal state D_t of StreamAlg, and the random bits R_t.

The goal of Adversary is to make StreamAlg output an incorrect response A_t to the query Q at some time $t \in [m]$ throughout the stream. By nature of the game, only a single pass over the stream is permitted.

Applications of white-box adversaries. The white-box adversarial model captures the ability of an adversary to adapt to internal processes of an algorithm which the black-box adversarial model is incapable of capturing. This property allows us to model richer adversarial scenarios. For example in the area of dynamic algorithms, the goal is to maintain a data structure that always outputs a correct answer at all times $t \in [m]$ across updates $u_1, \ldots, u_m$ that arrive sequentially, while minimizing either the overall running time or the worst-case update time. In some settings, the dynamic model also places a premium on space so that algorithms must use space sublinear in the size of the input, but generally this may not be required. The dynamic model often considers an adaptive adversary [22, 23, 60, 68] that generates the updates $u_1, \ldots, u_m$ upon seeing the entire data structure maintained by the algorithm after the previous update, i.e., a white-box adversary.

The algorithm's internal state can be used as part of a procedure that will ultimately generate future inputs. For example, consider a distributed streaming setting where a centralized server wants to collect statistics on a database generated by a number of remote users. The centralized server wants to minimize its space usage and therefore does not want to store each update by the remote users. Moreover, the server may want to limit communication over the network and thus it sends components of its internal state S (such as initialized random variables) to the remote users in order to optimize the information sent from the remote users back to the centralized server. The remote users may use S in some process that ultimately affects how the data downstream is generated. Thus in this case, the future input data depends on (components) of the internal state S of the streaming algorithm of the central coordinator; this scenario is captured by the white-box adversarial model. Furthermore, one of the remote users could be malicious and would

like to use the state S to cause the central coordinator to fail. In this case, the data is not only dependent on S but also adversarially generated; this scenario is also captured by the white-box adversarial model. Finally, in the case that there is no central coordinator, the entire internal state may be stored on a cloud, which would be visible to all users.

The pan-private streaming model [30] lets the internal state of the algorithm be partially or completely revealed. This model is often motivated by distributed users such as hospitals, government agencies, or search engine providers. [30] notes that any data curator "can be pressured to permit data to be used for purposes other than that for which they are collected", including uses that may ultimately affect the distribution of future input data to the curator. In fact, [53] in PODS 2011 specifically considers the problem of counting distinct elements and detecting heavy-hitters on a data stream when the internal state of the algorithm is revealed, giving the motivating example of an insider manipulating traffic flow while trying to find flaws in a systems administration database that tracks user visit statistics. It could be argued that although the goal of [53] is just to preserve the privacy of the users, they should also consider the white-box adversarial model where the future inputs depend on previous information rather than their assumption that the input is independent of the internal information.

In machine learning, robust algorithms and adversarial attacks have captured the attention of recent research. In 2017, Google Brain organized a competition at NeurIPS 2017 for producing effective adversarial attacks, in which many of the most successful attacks used knowledge of the internal algorithmic parameters and training weights to minimize some loss function in a small neighborhood around the original input [15, 34, 65]. More recently, white-box attacks have generated adversarial inputs by modifying existing data in such a minor way that is almost imperceptible to the human eye, either in images [38, 65] or in the physical world [6, 46, 63]. However, the modified data results in an incorrect classification by a machine learning algorithm. As a result, a large body of recent literature has focused on adversarial robustness of machine learning models against white-box attacks, e.g., [26, 39, 47, 49, 50, 61, 66].

In persistent data structures, the goal is to provide version control to an evolving data structure while minimizing either the space or time to view each version, e.g., [29, 31, 42]. For example, the ability to quickly access previous versions of information stored in an online repository shared across multiple collaborators is an invaluable tool that many services already provide. Moreover, the internal persistent data structures used to provide version control may be accessible and thus visible to all users of the repository. These users may then update the persistent data structure in a manner that is not independent of the previous states.

1.1 Our Contributions

In this paper, we study the abilities and limitations of streaming algorithms robust to white-box adversaries, which are significantly more powerful than black-box adversaries. An insightful example of this is the work [36] which develops a sophisticated attack for a black-box adversary to iteratively learn the matrix used for a linear sketch in the black-box adversarial model. On the other hand, the white-box adversary immediately sees the sketching matrix when the algorithm is initiated. More generally, techniques such as differential privacy, which are widely employed in black-box adversarial settings to hide internal information, will not work against white-box adversaries.

The main contributions of this paper can be summarized as introducing general tools to design algorithms robust to white-box adversaries as well as presenting a framework for proving strong lower bounds in the white-box adversarial model. In more detail, our contributions are the following:

- **White-box adversarial streaming model**: We introduce and formalize the white-box adversarial model for data streams as a means of modeling richer adversarial settings found in a wide variety of application areas which are not captured by the black-box adversarial model.
- **Robust algorithms and diverse applications**: We provide streaming algorithms robust against white-box adversaries for problems across many different domains, e.g., statistical problems such as heavy-hitters, graph algorithms, applications in numerical linear algebra, and string algorithms, with wide-ranging applications. For example, estimating F_p moments has applications in databases, computer networks, data mining, and other contexts such as in determining data skewness, which is important in parallel database applications [28] or determining the output of self-joins in databases [33]. L_0 estimation is used by query optimizers to find the number of unique values of some attribute without having to perform an expensive sort. This statistic is further useful for selecting a minimum-cost query plan [62], database design [48], OLAP [59, 64], data integration [18, 27], data warehousing [1], and packet tracing and database auditing [25]. For more details, see Section 1.1.1.
- **Use of cryptography in robust algorithms**: A key toolkit we widely employ to design our robust algorithms comes from cryptography. Leveraging computational assumptions commonly used for the design of cryptographic protocols allows us to use powerful algorithmic tools such as collision resistant hash functions and sketching matrices for which it is computationally hard to find a "short" vector in their kernel. We believe our work opens up the possibility of using cryptography much more broadly for streaming algorithms, beyond the white-box adversarial setting.
- **A general lower bound framework**: We give a general reduction from two-player communication problems to space lower bounds in the white-box adversarial model. Corollaries of our reduction include lower bounds for F_p moment estimation in data streams. For more details, see Section 3.1.
- **Lower bounds for deterministic counting**: Lastly, we provide a space lower bound for deterministic algorithms which count the number of ones in a binary stream in the oblivious model, even if the algorithm has access to a timer that reports how many total stream updates it has seen so far. This lower bound serves two purposes. First, it shows that our general lower bound framework of Section 3.1 does not extend to multiparty (greater than two) communication protocols. Second, it provides strong lower bounds for the fundamental problem of approximately counting in a stream, which has wide applications (see Section 3.2).

1.1.1 Robust Algorithms. Any deterministic algorithm is naturally robust in the white-box adversarial streaming model, but deterministic algorithms are often inefficient in a stream. We first show that the ε-L_1-heavy hitters problem can be solved using space strictly less than that of any deterministic algorithm in the white-box adversarial streaming model. In this problem, the goal is to output all indices i such that $f_i > \varepsilon L_1$, where $L_1 = \|f\|_1$ is the ℓ_1 norm of the underlying freqeuncy vector defined by the stream.

Theorem 1.1. *There exists a white-box adversarially robust algorithm that reports all ε-L_1-heavy hitters with probability at least $3/4$ and uses space $O\left(\frac{1}{\varepsilon}\left(\log n + \log\frac{1}{\varepsilon}\right) + \log\log m\right)$.*

By comparison, the well-known Misra-Gries data structure [55] is a deterministic algorithm and thus robust against white-box adversaries, but uses $O\left(\frac{1}{\varepsilon}(\log m + \log n)\right)$ bits of space. Our algorithm offers similar guarantees to Misra-Gries in the sense that it returns a list of $O\left(\frac{1}{\varepsilon}\right)$ items that contains all ε-L_1-heavy hitters as well as an approximate frequency for each item in the list with additive error εL_1. Note that in the ε-L_1-heavy hitters problem, the gap between the frequencies of items that appear in the list can be as large as $\Omega(\varepsilon) \cdot L_1$. On the other hand, the (φ, ε)-L_1-heavy hitter problem demands that we find all φ-L_1-heavy hitters, but report no items whose frequency is below $(\varphi - \varepsilon) L_1$, thereby parametrizing the threshold of the "false positives" that are reported in the list. We give an algorithm with the following guarantees for the (φ, ε)-L_1-heavy hitter problem against white-box adversaries with total runtime T, i.e., T-time bounded adversaries:

Theorem 1.2. *There exists an algorithm robust against white-box T-time bounded adversaries that solves the (φ, ε)-L_1-heavy hitter problem with probability at least $3/4$ and uses total space $O\left(\frac{1}{\varepsilon}\left(\log\log n + \log\frac{1}{\varepsilon}\right) + \frac{1}{\varphi}\log n + \log\log m\right)$ for $T < \text{poly}\left(\log n, \frac{1}{\varepsilon}\right)$ and total space $O\left(\frac{1}{\varepsilon}\min(\log n, \log T) + \frac{1}{\varphi}\log n + \log\log m\right)$ for $T \geq \text{poly}\left(\log n, \frac{1}{\varepsilon}\right)$.*

We remark that the proof of Theorem 1.2 uses collision-resistant hash functions and hence only guarantees robustness against white-box adversaries with polynomially bounded computation time. Theorem 1.2 is not information-theoretically secure against white-box adversaries with unbounded computation time.

We obtain qualitatively similar results to that of Theorem 1.1 for the *Hierarchical Heavy Hitters* problem, which generalizes L_1-heavy-hitters; see the full version of the paper for more details. We also obtain similar results for the vertex neighborhood identification problem, where the task is for an algorithm to identify all vertices of a graph with identical neighborhoods, in the vertex arrival model, where each update of the stream is a vertex of the graph along with a list of all of its neighbors.

Theorem 1.3. *There exists an algorithm robust against white-box polynomial-time adversaries that reports all vertices with identical neighborhoods with probability at least $3/4$, using space $O(n \log n)$.*

We also prove a lower bound for deterministic algorithms for the vertex neighborhood identification problem, even on oblivious data streams:

Theorem 1.4. *Any deterministic algorithm that reports all vertices with identical neighborhoods uses space $\Omega\left(\frac{n^2}{\log n}\right)$.*

Together, Theorem 1.3 and Theorem 1.4 show a strong separation between deterministic algorithms and randomized algorithms robust against white-box adversaries with polynomial runtime. We further utilize cryptographic tools to obtain robust algorithms for other fundamental streaming problems. In particular, assuming the hardness of the 'Short Integer Solution' (SIS) problem of lattice cryptography (see Definition 2.7 and Theorem 2.8), we can obtain a sublinear space algorithm for the L_0 estimation problem:

Theorem 1.5. *Let $c \in (0, 1/2)$ and assume the adversary cannot solve the SIS problem of Definition 2.7 with parameter $\beta_\infty = \text{poly}(n)$ in Theorem 2.8 for sufficiently large n. Then Algorithm 3 returns a n^ε multiplicative approximation to the L_0 estimation problem. Furthermore, the algorithm uses space $n^{1-\varepsilon+c\varepsilon} + n^{(1+c)\varepsilon}$. In the random oracle model, the algorithm uses space $n^{1-\varepsilon+c\varepsilon}$.*

Note that the above result also guarantees robustness against a computationally bounded adversary, formalized in Assumption 2.9. With no such assumptions, we can provably show that approximating general F_p moments up to constant factors in data streams is impossible in sublinear space (see Theorem 1.9 below). Using the same cryptographic assumption, we also obtain streaming algorithms for the rank-decision problem in linear algebra.

Theorem 1.6. *Suppose A is a $n \times n$ matrix. There exists a constant $0 < c < 1$ such that the rank decision problem for A can be solved in $\tilde{O}\left(nk^2\right)$ bits of space for any $k \leq n^c$ under the random oracle model assuming a computationally bounded adversary.*

Corollaries of this result include streaming algorithms for other linear algebra based applications such as computing a linearly independent basis. Lastly, we also obtain white-box robust algorithms for string pattern matching applications.

Theorem 1.7. *For an input string P with given period p, followed by a string U, there exists a streaming algorithm that, with probability at least $1 - \frac{1}{\text{poly}(n)}$, finds all instances of P within U. This streaming algorithm is robust against T-time white-box adversaries and uses $O(\log T)$ bits of space.*

1.1.2 A General Reduction for Lower Bounds. A natural question is whether there exist robust streaming algorithms against white-box adversaries for more complex problems, such as F_p estimation or matrix rank. To that end, we give a general technique to prove lower bounds for *randomized* algorithms robust to white-box algorithms through two-player *deterministic* communication problems.

Theorem 1.8. *(Informal) Suppose there exists a white-box adversarially robust streaming algorithm using $S(n, \varepsilon)$ space that can be used to solve a one-way two-player communication game with $S(n, \varepsilon)$ bits of communication with probability $p \in (1/2, 1]$. Then there exists a deterministic protocol for the two-player communication game using $S(n, \varepsilon)$ bits of communication.*

We remark that Theorem 1.8 is especially powerful because it can be used to prove lower bounds for randomized algorithms robust against white-box adversaries using reductions from deterministic

communication problems, which can often have much higher communication complexity than their randomized counterparts. For example, the deterministic complexity of the Equality problem, in which Alice must send Bob a message to determine whether their strings $x \in \{0,1\}^n$ and $y \in \{0,1\}^n$ are equal, is $\Theta(n)$. However, the randomized complexity of the Equality problem is $\Theta(\log n)$ for a constant probability of success. Thus Theorem 1.8 can show significantly stronger lower bounds for randomized algorithms robust against white-box adversaries over the standard streaming model. In particular, we first obtain the following hardness of approximation for F_p moment estimation from Theorem 1.8:

Theorem 1.9. *For each $p \geq 0$ and $p \neq 1$, there exists a constant $C_p > 1$ such that any white-box adversarially robust algorithm that reports a C_p-approximation to the frequency moment F_p of an underlying frequency vector with probability at least $2/3$ must use space $\Omega(n)$.*

We also obtain the following hardness of approximation for matrix rank estimation from Theorem 1.8:

Theorem 1.10. *There exists a constant $C > 1$ such that any white-box adversarially robust algorithm that reports a C-approximation to the rank of an underlying matrix with probability at least $2/3$ must use space $\Omega(n)$.*

1.1.3 Lower Bound for Deterministic Counting. The "basic" problem of counting the number of ones in a binary stream, or equivalently the number of stream updates, is arguably the most fundamental streaming problem with both practical and theoretical applications. While there exists a straightforward $O(\log n)$ space algorithm that explicitly maintains the actual count over the stream, there exist randomized algorithms, notably the Morris counters, which achieve $O(\log \log n)$ bits. This savings is particularly meaningful when considering applications such as counting the number of visits to a popular website such as Wikipedia. In these applications, it is common to maintain many counters rather than one, and thus optimizing the space usage per counter has a measurable overall impact.

The basic counting problem is a key subroutine in many streaming problems in the oblivious model, such as F_p estimation in an insertion-only stream [41], approximate reservoir sampling [35], approximating the number of inversions in a permutation [3], and L_1-heavy hitters in insertion streams [14].

We show that any deterministic problem in the oblivious model for counting must asymptotically use the same amount of space as the trivial algorithm, even when the algorithm knows the identity of the current position in the stream. That is, even if the streaming algorithm is augmented with a "timer" which tells it at any time how many stream updates it has seen, the algorithm still cannot approximate the number of 1s in a binary stream up to a constant factor unless it uses $\Omega(\log n)$ bits of memory. Note that having a timer is what makes our lower bound nontrivial: without a timer, with $o(\log n)$ bits of memory there are fewer than say, $n/10$ states of the algorithm, so after seeing $n/10$ 1's, the algorithm necessarily revisits a state it has seen before. Since the algorithm is deterministic, it gets stuck in a cycle and thus can produce at best a 10-approximation. A timer will also be useful for our application, described momentarily.

Theorem 1.11. *Given a constant $\varepsilon > 0$, any deterministic algorithm that outputs a $(1+\varepsilon)$-approximation to the number of ones in a length-n stream of bits must use $\Omega(\log n)$ bits of space, even if the algorithm has a timer which tells it how many stream updates it has seen so far.*

Surprisingly, Theorem 1.11 shows that our technique translating two-player deterministic communication lower bounds to white-box adversary lower bounds in Theorem 1.8 cannot extend to an arbitrary number n of players. Recall that Theorem 1.8 implies the white-box space complexity is at least the two-player one-way deterministic communication of the underlying communication problem, which is just the maximum communication of any player (since only one player speaks). A natural question is whether the white-box space complexity is at least the maximum communication of the underlying n-player deterministic communication game. This is false, since in the white-box adversarial model we can count using Morris counters with $O(\log \log n)$ bits. However, given Theorem 1.11, the maximum communication of the underlying n-player deterministic communication game is $\Omega(\log n)$ bits. Note that in a communication game, each player knows its identity and can behave differently than other players, and thus the assumption that the algorithm has a timer in Theorem 1.11 is needed.

1.2 Overview of our Techniques

L_1-heavy hitters. To find all ε-L_1-heavy hitters, we first recall that the well-known Misra-Gries algorithm is deterministic and maintains approximate frequencies to each of the possible heavy hitters using space $O\left(\frac{1}{\varepsilon}(\log m + \log n)\right)$, which is expensive for $m \gg 2^n$. Thus if we can reduce the stream length from m to some $m' = \text{poly}\left(\frac{1}{\varepsilon}, n\right)$, then we can run Misra-Gries on the smaller stream. If the stream length m were known, we can use Bernoulli sampling on each update in the stream with probability roughly $\frac{\log n}{\varepsilon^2 m}$ to preserve the ε-L_1-heavy hitters; this sampling probability was shown to be secure against a white-box adversary in [13]. Thus it remains to resolve the issue of not knowing the stream length m in advance.

A natural approach is to make a number of exponentially increasing guesses for the length of the stream m. However, not only does this approach induce a multiplicative overhead of $O(\log n)$ in the number of simultaneous instances, corresponding to each guess for the length, but also even tracking the length m of the stream exactly requires $O(\log m)$ bits, which we would like to avoid. We instead observe that Morris counters are white-box adversarially robust and use them to estimate the length of the stream at all times within a constant factor with space $O(\log \log m)$. Moreover, we simultaneously only maintain two guesses for the length of the stream, corresponding to increasing powers of $\left(\frac{16}{\varepsilon}\right)$, since when an instance of the algorithm is initiated with a guess for m, at most $\frac{\varepsilon}{16} m$ updates have been missed by the algorithm, so any items i that are ε-L_1-heavy hitters of the stream will still be $O(\varepsilon)$-L_1-heavy hitters of the stream seen by the algorithm. Similar techniques also work for the Hierarchical Heavy Hitters problem.

Computationally-bounded white-box adversaries. We can further improve our guarantees if the white-box adversary has bounded

computational time through the use of collision-resistant hash functions. Namely, we can solve the (φ, ε)-L_1-heavy hitters problem by using a collision-resistant hash function to hash the sampled items into a universe of size poly $\left(\log n, \frac{1}{\varepsilon}, T\right)$ for white-box adversaries with computation time at most T. Similarly, we can achieve a streaming algorithm for the vertex neighborhood identification problem that is robust against polynomial-time white-box adversaries by hashing a Boolean vector representing the neighborhood of each vector into a universe of size poly(n, T). Thus it suffices to maintain n hashes corresponding to the n neighborhoods, using total space $O(n \log nT)$. By comparison, we can use the OR Equality problem to show any deterministic algorithm that solves the vertex neighborhood identification problem requires $\Omega\left(\frac{n^2}{\log n}\right)$ space. We similarly use collision-resistant hash functions to obtain robust algorithms for problems in linear algebra and strings.

Two-player communication lower bounds. To acquire a tool for proving lower bounds for white-box adversarially robust algorithms, we first note that a standard technique for proving lower bounds in the oblivious streaming model is to consider the randomized communication complexity of certain problems, such as Equality. However, the deterministic communication complexity of many of these problems can be a lot higher. Surprisingly, we show that randomized algorithms that are robust against white-box adversaries can be used for deterministic protocols, thereby proving significantly stronger lower bounds for white-box adversarially robust algorithms than their counterparts in the oblivious streaming model. In particular, if there exists a randomized algorithm that is robust against white-box adversaries, then it can be used in a two-player communication game as follows. The first player reads their input and creates the stream as usual. Rather than generating internal randomness for the algorithm, the first player notes that there exists a choice of the internal randomness such that for any possible input to the second player, the algorithm succeeds on 9/10 of the possible strings used for randomness by the second player. The first player can then enumerate over all possible inputs and all possible strings used for the randomness to the second player and choose the first internal randomness for the first player such that the guarantee holds. The first player can then run the algorithm on the created stream with this *deterministic* choice of randomness and pass the state to the second player, who can now run their input over all choices of randomness for the second part of the stream, and take a majority vote. This results in a deterministic protocol and thus must respect any deterministic lower bound for a communication problem. In particular, we can choose the communication problem to be Gap Equality (which is just the Equality problem with the promise that when the two input strings are not equal, they differ in a constant fraction of positions) problem to show both Theorem 1.9 and Theorem 1.10. Unfortunately, we prove that this technique cannot be generalized to show lower bounds for white-box adversarially robust algorithms through multiplayer communication.

Lower Bounds for Deterministic Counting with a Timer. A streaming algorithm is just a read-once branching program. Theorem 1.11 is proven by bounding the number of counts that a single state of the branching program can correctly represent, which translates to an upper bound on the length of an interval on a worst-case stream that each state can correctly represent. We then show that there exists some t_0 such that after t_0 updates in the stream, all read-once branching programs require at least poly(n) states to approximate the number of ones in a length n stream to within a constant factor. See Section 3.2 for a more detailed description.

1.3 Notation

For an integer $n > 0$, $[n]$ denotes the set of integers $\{1, \ldots, n\}$. We use poly(n) to denote a fixed constant degree polynomial in n and $\frac{1}{\text{poly}(n)}$ to denote an arbitrary degree polynomial in n that can be determined from setting constants appropriately. For a vector $v \in \mathbb{R}^n$, we use v_k with $k \in [n]$ to denote its k-th coordinate. Given vectors $u, v \in \mathbb{R}^n$, we write their inner product as $\langle u, v \rangle = \sum_{k=1}^{n} u_k v_k$. For a string S, we use $S[i : j]$ to denote the substring formed from the i-th character of S to the j-th character of S, inclusive.

2 UPPER BOUNDS

In this section, we present streaming algorithms relevant to statistics that are robust to white-box adversaries. All missing proofs in this paper can be found in Section A.

We first describe a crucial data structure for our algorithms: the Morris counter [56]. This is a data structure for the approximate counting problem, where a nonnegative integer $Z = \sum_{t=1}^{m} u_t$ is defined through updates $u_1, \ldots, u_m$ such that $u_t \in \{0, 1\}$ for each $t \in [m]$. Given an accuracy parameter $\varepsilon > 0$, the goal of the approximate counting problem is to estimate Z to within a $(1 + \varepsilon)$-approximation. Our first result is that Morris counters are robust in the white-box adversarial model.

Lemma 2.1. *Morris counters output a $(1 + \varepsilon)$-approximation to the frequency of an item i in the white-box adversarial streaming model with probability at least $1 - \delta$, using total space*

$$O\left(\log\log n + \log\frac{1}{\varepsilon} + \log\log m + \log\frac{1}{\delta}\right).$$

2.1 Heavy-Hitters

In this section, we present randomized algorithms for ε-L_1-heavy hitters that provide better guarantees than the well-known deterministic Misra-Gries data structure. In the F_p moment estimation and L_p norm estimation problems, an underlying frequency vector $f \in \mathbb{R}^n$ is defined through updates $u_1, \ldots, u_m$ such that $u_t \in [n]$ for each $t \in [m]$. The resulting frequency vector f is then defined so that $f_k = |\{t \mid u_t = k\}|$ for each $k \in [n]$. For $p > 0$, the F_p moment of f is defined to be $F_p(f) = \sum_{i=1}^{n} (f_k)^p$. The L_p norm of f is $\|f\|_p = (F_p(f))^{1/p}$. We use both F_0 and L_0 to denote the number of nonzero coordinates of f, i.e., $F_0(f) = \|f\|_0 = |\{k \mid f_k \neq 0\}|$. Given a threshold parameter $\varepsilon > 0$, the goal of the F_p moment estimation problem is to provide a $(1+\varepsilon)$-approximation to $F_p(f)$ and the goal of the ε-L_p-heavy hitters problem is to find all coordinates k such that $f_k \geq \varepsilon L_p(f)$. In the (φ, ε)-L_p-heavy hitter problem, the goal is to report all coordinates k such that $f_k \geq \varphi L_p(f)$ but also no coordinate j such that $f_j \leq (\varphi - \varepsilon) L_p(f)$.

Theorem 2.2. *[55] Given a threshold parameter $\varepsilon > 0$, there exists a deterministic one-pass streaming algorithm MisraGries that*

uses $O\left(\frac{1}{\varepsilon}(\log m + \log n)\right)$ bits of space on a stream of length m and outputs a list L of size $\frac{1}{\varepsilon}$ that includes all items i such that $f_i > \varepsilon m$. Moreover, the algorithm returns an estimate $\widehat{f_i}$ for each $i \in L$ such that $f_i - \varepsilon m \le \widehat{f_i} \le f_i$.

To output the identities of $\frac{1}{\varepsilon}$ heavy-hitters, $\Omega\left(\frac{1}{\varepsilon}\log n\right)$ space is clearly necessary. On the other hand, it is not clear that the dependence on $\log m$ for Misra-Gries in Theorem 2.2 is needed. It is known that we can essentially preserve the ε-L_1-heavy hitters by sampling a small number of updates in a stream.

Theorem 2.3. *[13] There exists a constant $C > 0$ such that for any $\varepsilon, \delta \in (0, 1/2)$, universe size n, and stream length m, Bernoulli sampling each item of the stream with probability $p \ge \frac{C\log(n/\delta)}{\varepsilon^2 m}$ solves the heavy hitters problem with error ε in the white-box adversarial model.*

We remark that Theorem 2.3 was proven in [13] against black-box adversaries, but their proof extends naturally to white-box adversaries because there is no additional private randomness maintained by the algorithm. On the other hand, Theorem 2.3 requires that the length of the stream is known a priori, which is an assumption that we would like to remove by running multiple instances of the algorithm in parallel, with exponentially increasing guesses for the length of the stream. However, even to track the length of the stream requires $O(\log m)$ bits of space. Instead, only an approximation to the length of the stream is required and thus we use Morris counters to remove the dependence on $\log m$. Finally, we observe that it suffices to maintain only two active guesses for the length of the stream at any point in time because even if we start a guess "late" in the stream, we will have only missed a $\mathrm{poly}(\varepsilon)$-prefix length of the stream, so any ε-L_1-heavy hitters will still be $O(\varepsilon)$-heavy with respect to the substream.

A useful subroutine we will need appears in Algorithm 1, and our full algorithm appears in Algorithm 2. Algorithm 2 gives the full guarantees of Theorem 1.1. We can also generalize this approach to hierarchical heavy hitters; we give more details in the full version of the paper.

Algorithm 1 BernMG$(n, m, \varepsilon, \delta)$

Input: Universe size n, upper bound m on the stream length, accuracy ε, failure probability δ, and a stream of updates $u_1, u_2, \ldots$, where each $u_i \in [n]$ represents a single update to a coordinate of the underlying vector f

Output: ε-L_1-heavy hitters of the stream

1: Initialize an instance $\mathcal{A}$ of Misra-Gries with threshold $\frac{\varepsilon}{2}$.
2: **for** each update u_t with $t \in [m]$ **do**
3: With probability $O\left(\frac{\log(n/\delta)}{\varepsilon^2 m}\right)$, update $\mathcal{A}$ with u_t
4: **return** the output of $\mathcal{A}$

Algorithm 2 also solves the (φ, ε)-L_1-heavy hitter problem in which all items i such that $f_i \ge \varphi\|f\|_1$ are outputted and no items j such that $f_j < (\varphi - \varepsilon)\|f\|_1$ are outputted, using a total space of $O\left(\frac{1}{\varepsilon}\left(\log n + \log\frac{1}{\varepsilon}\right) + \log\log m\right)$ bits. [14] showed that for oblivious streams, the $O\left(\frac{1}{\varepsilon}\log n\right)$ dependence is not necessary. Similarly,

Algorithm 2 Adversarially robust algorithm for ε-L_1-heavy hitters

Input: Universe size n, accuracy ε, and a stream of updates $u_1, u_2, \ldots$, where each $u_i \in [n]$ represents a single update to a coordinate of the underlying vector f

Output: ε-L_1-heavy hitters of the stream

1: Run a Morris counter that outputs a $(1 + O(\varepsilon))$-approximation $\widehat{t}$ to the number of stream updates $t \in [m]$.
2: $c \leftarrow 0$, $r \leftarrow 2$, $\delta \leftarrow O\left(\frac{\varepsilon}{\log m}\right)$
3: **for** $i \in [r]$ **do**
4: Initialize an instance $\mathcal{A}_i$ of BernMG$(n, (16/\varepsilon)^i, \varepsilon/2, \delta)$
5: **for** each update u_t with $t \in [m]$ **do**
6: Update all instances of $\mathcal{A}_i$
7: **if** $\widehat{t} \ge (16/\varepsilon)^c$ **then**
8: Delete $\mathcal{A}_c$
9: $c \leftarrow c + 1$
10: Initialize an instance $\mathcal{A}_c$ of
11: BernMG$(n, (16/\varepsilon)^{c+1}, \varepsilon/2, \delta)$
12: **return** the output of $\mathcal{A}_c$

we can further improve our bounds against white-box adversaries that use $T = \mathrm{poly}(\kappa)$ time through the following notion of collision-resistant hash functions:

Definition 2.4 (Family of Collision-Resistant Hash Functions)**.** *A set of functions $H = \{h_i : \{0,1\}^{n_i} \to \{0,1\}^{m_i}\}_{i \in I}$ is a family of collision-resistant hash functions (CRHF) if*

- *(Efficient generation) There exists a probabilistic polynomial-time algorithm* Gen *such that* $\mathsf{Gen}(1^\kappa) \in I$ *for all* $\kappa \in \mathbb{Z}^+$.
- *(Compression)* $m_i < n_i$ *for all* $i \in I$.
- *(Efficient evaluation) There exists a probabilistic polynomial-time algorithm* Eval *such that for all* $i \in I$ *and* $x \in \{0,1\}^n_i$, *we have* $\mathsf{Eval}(x, i) = h_i(x)$.
- *(Collision-resistant) For any non-uniform probabilistic polynomial-time algorithm $\mathcal{A}$, there exists a negligible function* negl *such that for all security parameters* $\kappa \in \mathbb{N}$,

$$\Pr_{(x_0,x_1)\leftarrow\mathcal{A}(1^\kappa,h)}[x_0 \ne x_1 \wedge h(x_0) = h(x_1)] \le \mathrm{negl}(\kappa).$$

Here we use negligible function to mean a function negl such that $\mathrm{negl}(x) = o(1/x^c)$ for any constant $c > 0$. There are folklore constructions of collision-resistant hash functions based on the hardness of finding the discrete logarithm of a given composite number, e.g., Theorem 7.73 in [44]:

Theorem 2.5. *Under the discrete log assumption, there exists a family of collision-resistant hash functions with $m_i = O(\log\kappa)$ for $i = \mathsf{Gen}(1^\kappa)$ and uses $O(\log\kappa)$ bits of storage.*

Theorem 1.2 then follows from applying Theorem 2.5 to the sampled items. We also remark that Algorithm 2 can be used to estimate the inner product of two vectors f and g that are implicitly defined through two streams:

Corollary 2.6. *There exists a white-box adversarially robust algorithm that uses space $O\left(\frac{1}{\varepsilon}\left(\log n + \log\frac{1}{\varepsilon}\right) + \log\log m\right)$ and outputs vectors $f', g' \in \mathbb{R}^n$ such that with probability at least $3/4$,*

$$|\langle f', g'\rangle - \langle f, g\rangle| \le \varepsilon\|f\|_1\|g\|_1.$$

2.2 L_0 Estimation

We now provide a streaming algorithm for the L_0 estimation problem, where the goal is to estimate the number of nonzero coordinates at the end of the stream. From the lower bound of Theorem 1.9, we cannot hope to estimate the L_0 norm of f arbitrarily well. Surprisingly, we can attain a a multiplicative approximation of n^ε for arbitrarily small ε even in the turnstile setting if we assume a *computationally bounded* adversary, similar to assumptions made in cryptography. Our model of a computationally bounded adversary will deal with the following SIS problem from lattice based cryptography.

Definition 2.7 (Short Integer Solution (SIS) Problem)**.** *Let n, m, q be integers and let $\beta > 0$. Given a uniformly random matrix $A \in \mathbb{Z}_q^{n\times m}$, the SIS problem is to find a nonzero integer vector $z \in \mathbb{Z}^m$ such that $Az = 0 \bmod q$ and $\|z\|_2 \le \beta$.*

Starting from Ajtai's work [2], it is known that the SIS problem enjoys an average-case to worst-case hardness. That is, for some appropriate parameter settings, solving the SIS problem is at least as hard as approximating several fundamental lattice based cryptography problems in the worst case.

Theorem 2.8. *[52] Let n and $m = \text{poly}(n)$ be integers, let $\beta \ge \beta_\infty \ge 1$ be reals, let $Z = \{z \in \mathbb{Z}^m : \|z\|_2 \le \beta \text{ and } \|z\|_\infty \le \beta_\infty\}$, and let $q \ge \beta \cdot n^\delta$ for some constant $\delta > 0$. Then solving SIS on average with non-negligible probability, and with parameters n, m, q and solution set $Z \setminus \{0\}$, is at least as hard as approximating lattice problems in the worst case on n-dimensional lattices to within a factor of $\gamma = \max\{1, \beta \cdot \beta_\infty/q\} \cdot \tilde{O}\left(\beta\sqrt{n}\right)$.*

In the cryptography literature, lattice-based cryptography schemes are designed for any γ smaller than $2^{o(n \log\log n/\log n)}$ and the best approximation currently known is for $\gamma = 2^{O(n \log\log n/\log n)}$ via the LLL algorithm [67]. Improving the approximation factor to any asymptotically smaller γ would be a major breakthrough in cryptography. Therefore, our computational assumption is the following, which implies breaking any of our algorithms would require a major cryptographic breakthrough:

Assumption 2.9. *No adversary can approximate worst-case n-dimensional lattice problems within a $\gamma = 2^{o(n(\log\log n)/\log n)}$ factor.*

In the L_0 streaming algorithm, we will only rely on hardness for much smaller values of γ. Our algorithm for the L_0 estimation problem in data streams is the following. It first considers a partition of $[n]$ into $n^{1-\varepsilon}$ consecutive chunks each of n^ε coordinates. It then keeps track of $n^{1-\varepsilon}$ vectors, one for each chunk, by multiplying the corresponding update with a sketching matrix derived from the SIS problem. Our final estimate is the number of our $n^{1-\varepsilon}$ sketches which are nonzero when the stream ends. Note that we use the same sketching matrix A on each chunk, as described below.

We now claim that if the final frequency vector f satisfies $\|f\|_\infty \le \text{poly}(n)$, then we can achieve an n^ε multiplicative approximation for the L_0 estimation problem. Furthermore, we can achieve improved sublinear space if we are working in the random oracle model of cryptography, which was introduced in the pioneering work of Bellare and Rogaway [9]. In the random oracle model, we assume a publicly accessible random function which can be accessed to us *and* the adversary. Each query gives a uniform random value from some output domain and repeated queries give consistent answers. The random oracle model is a well-studied model and has been used to design numerous cryptosystems [9, 10, 20, 45]. In practice, one can use SHA256 as the random oracle.

Algorithm 3 Estimate-L0(n, m, ε)

Input: Universe size n, accuracy ε, and a stream of updates $u_1, u_2, \ldots$, where each $u_i \in [n]$ represents a single update to a coordinate of the underlying vector f, and each u_i is an integer

Output: n^ε-multiplicative estimation of L_0 of f

1: Consider $A \in \mathbb{Z}_q^{n^{c\varepsilon}\times n^\varepsilon}$ is a uniformly random matrix for $q = \text{poly}(n)$ and any $1/2 > c > 0$
2: Keep track of $n^{1-\varepsilon}$ vectors of length $n^{c\varepsilon}$, initially all 0 and each associated with a specific consecutive chunk of n^ε coordinates of $[n]$
3: **for** each update u_t with $t \in [m]$ **do**
4: Update the sketch vector associated with the i-th chunk by adding $u_t \cdot A_k$ to it, where A_k is the k-th column of A, and where the stream update changes the k-th coordinate of the i-th chunk by an additive amount $u_t \in \mathbb{Z}$
5: **return** the number of vectors that are nonzero

Theorem 1.5. *Let $c \in (0, 1/2)$ and assume the adversary cannot solve the SIS problem of Definition 2.7 with parameter $\beta_\infty = \text{poly}(n)$ in Theorem 2.8 for sufficiently large n. Then Algorithm 3 returns a n^ε multiplicative approximation to the L_0 estimation problem. Furthermore, the algorithm uses space $n^{1-\varepsilon+c\varepsilon} + n^{(1+c)\varepsilon}$. In the random oracle model, the algorithm uses space $n^{1-\varepsilon+c\varepsilon}$.*

We remark that we only require $\gamma = \text{poly}(n)$ for the application of Theorem 2.8 to Theorem 1.5. Furthermore, the algorithm also works for **turnstile** streams where the stream updates are allowed to be positive and negative. This is because we only require the final frequency vector f to satisfy $\|f\|_\infty \le \text{poly}(n)$ in Theorem 2.8; the signs of the entries in f do not matter.

2.3 Graphs and Linear Algebra Applications

We also consider graph and linear algebra applications in the white-box adversarial streaming model. We show that the neighborhood identification problem, in which the goal is to identify vertices whose neighborhoods are identical, can be solved in $O\left(n \log(nT)\right)$ space in the vertex arrival model via a randomized algorithm (Theorem 1.3). We show that the space bound is tight and furthermore that any deterministic algorithm must use space $\Omega(n^2/\log n)$ (Theorem 1.4). See the full version of the paper for more details.

We also consider the rank decision problem in linear algebra: given a stream of rows of a $n \times n$ matrix and a parameter k, the goal is to determine if the matrix has rank smaller than k or at least k. Assuming the hardness of the SIS problem (Theorem 2.8), we give a white-box adversarially robust algorithm using $\tilde{O}\left(nk^2\right)$ bits of space for the rank estimation problem (Theorem 1.6). Corollaries of Theorem 1.6 include recovering a linearly independent basis in a stream. See the full version of the paper for more details.

3 LOWER BOUND TECHNIQUES

In this section, we present techniques to showing lower bounds in the white-box adversarial streaming model. Surprisingly, our reductions can utilize deterministic communication complexity protocols, even to show lower bounds for randomized algorithms. We apply our techniques to show lower bounds for F_p estimation for all $p \geq 0$, including estimating the number of distinct elements for $p = 0$, as well as matrix rank estimation.

3.1 Lower Bounds for F_p Estimation

We prove a general reduction from two-player communication problems, i.e., Theorem 1.8, in Appendix A.2. We illustrate Theorem 1.8 with a lower bound for F_p moment estimation through a reduction from the following formulation of the Gap Equality problem:

Definition 3.1 (Gap Equality)**.** *In the deterministic Gap Equality problem* DetGapEQ$_n$*, Alice receives a string $x \in \{0,1\}^n$ with $|x| = \frac{n}{2}$ and Bob receives a string $y \in \{0,1\}^n$ with $|y| = \frac{n}{2}$. Their goal is to use a deterministic protocol to determine whether $x = y$, given the promise that either $x = y$ or the Hamming distance* $\mathsf{HAM}(x, y)$ *satisfies* $\mathsf{HAM}(x, y) \geq \frac{n}{10}$.

Theorem 3.2. *([19]) The deterministic communication complexity of* DetGapEQ$_n$ *is $\Omega(n)$.*

We use the Gap Equality problem to show lower bounds for F_p moment estimation in the white-box adversarial streaming model; Theorem 1.10 follows from a similar reduction. We remark that a stronger version can be proved, in which the streaming algorithm can hide bits from the adversary.

Theorem 3.3. *For any $p \geq 0$ with $p \neq 1$, there exists a constant $C_p > 1$ such that any white-box adversarially robust algorithm that outputs an C_p-approximation to F_p with probability at least $9/10$ requires $\Omega(n)$ space.*

3.2 Lower Bound for Deterministic Approximate Counting

A natural question is whether the techniques of Theorem 1.8 extend to multiplayer communication games. Unfortunately, Theorem 1.11 shows that the technique provably cannot generalize. Theorem 1.11 is proven by showing that all read-once branching programs require at least $\text{poly}(n)$ states to approximately count the number of ones in a length n stream. Hence in a communication protocol across n players where each player is given a single bit and the goal is to approximately count the number of ones held across all players, the maximum communication by a single player must be at least $\Omega(\log n)$ bits. This implies that our techniques which reduce hardness for white-box adversarially robust algorithms to two-player deterministic communication lower bounds *cannot* be generalized to multiplayer deterministic communication lower bounds, since such a generalization would imply a space lower bound of $\Omega(\log n)$ for approximate counting, whereas Morris counters use $O\left(\log\log n + \log\frac{1}{\varepsilon} + \log\frac{1}{\delta}\right)$ bits of space.

3.3 A Communication Complexity Model for White-Box Adversaries

We formulate Theorem 1.8 in terms of a communication matrix that differs from existing two-player communication games because the protocol may not necessarily succeed against all possible inputs to the players; there can be specific inputs to the protocol that cause failure, provided that these inputs cannot be found by a T-time randomized algorithm. The communication model is particularly interesting due to the equality problem and its previously discussed complexity in this model, which depends on the runtime T. However, problems like set disjointness and index do not seem to exhibit such a dependence. Thus this communication complexity model may be of independent interest.

Assume that there exists a streaming algorithm $\mathcal{A}$ robust against T-time white-box adversaries that can be used to compute a function $f(x, y)$ using s bits of communication with probability p. In the communication protocol, Alice creates a stream S_x that induces the input x. Alice runs $\mathcal{A}$ on S_x and communicates the s-bit state of $\mathcal{A}$ to Bob. Bob then continues running $\mathcal{A}$ on a stream S_y that induces the input y, starting from the s-bit state that Bob receives from Alice. The output $f(x, y)$ is the output of $\mathcal{A}$ when run on the stream $S_x \circ S_y$, where $\circ$ denotes concatenation.

We now define a communication matrix for the randomized one-way communication protocol induced by $\mathcal{A}$. Consider a matrix M whose rows are indexed by tuples (x, r_x), where x is Alice's input and r_x is Alice's randomness, and whose columns are indexed by tuples (y, r_y), where y is Bob's input and r_y is Bob's randomness. The entry $M_{(x,r_x),(y,r_y)}$ of M denotes the output of the two-player communication game when Alice holds (x, r_x) and Bob holds (y, r_y). Since $\mathcal{A}$ uses s space, there exists a partition of the rows of M into 2^s parts such that if (x, r_x) and $(x', r_{x'})$ are in the same part, then $M_{(x,r_x),(y,r_y)} = M_{(x',r_{x'}),(y,r_y)}$. This corresponds to the fact that Alice sends Bob the same s-bit state $\text{state}(x, r_x)$ whether Alice held (x, r_x) or $(x', r_{x'})$. For each (x, r_x), define $p_{\text{state}(x,r_x)} = \min_y \Pr_{r_y}[M_{(x,r_x),(y,r_y)} = f(x, y)]$, which is the minimum probability that $\mathcal{A}$ outputs $f(x, y)$ over all possible inputs y chosen by a white-box adversary. Note that $p_{\text{state}(x,r_x)}$ is well-defined because $M_{(x,r_x),(y,r_y)} = M_{(x',r_{x'}),(y,r_y)}$ whenever $\text{state}(x, r_x) = \text{state}(x', r_{x'})$. By the robustness of $\mathcal{A}$ against a white-box adversary, we have the guarantee that for all inputs x, $\mathbb{E}_{r_x}[p_{\text{state}(x,r_x)}] \geq p$.

We now consider the situation in which the white-box adversary A is computationally bounded. In this setting, the adversary may not be able to enumerate over all inputs y and output the y that minimizes $\Pr_{r_y}[M_{(x,r_x),(y,r_y)} = f(x, y)]$. Hence a communication protocol $\mathcal{A}$ robust against computationally bounded white-box adversaries A satisfies $\mathbb{E}_{y=A(\text{state}(x,r_x))} \Pr_{r_y}[M_{(x,r_x),(y,r_y)} = f(x, y)] \geq p$ for all computationally bounded adversaries A and inputs x, which is a weaker guarantee.

ACKNOWLEDGEMENTS

S.S. and A.S. were supported by the NSF Graduate Research Fellowship under Grant Nos. 1745302, 1745016, and 2140739. D.P.W. and S.Z. were supported by a Simons Investigator Award and by the NSF Grant No. CCF-1815840. Prepared by LLNL under Contract DE-AC52-07NA27344. LLNL-CONF-833470.

REFERENCES

[1] Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. 1999. The Aqua Approximate Query Answering System. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data*. 574–576.

[2] Miklós Ajtai. 1996. Generating Hard Instances of Lattice Problems (Extended Abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*. 99–108.

[3] Miklós Ajtai, T. S. Jayram, Ravi Kumar, and D. Sivakumar. 2002. Approximate counting of inversions in a data stream. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing*. 370–379.

[4] Noga Alon, Omri Ben-Eliezer, Yuval Dagan, Shay Moran, Moni Naor, and Eylon Yogev. 2021. Adversarial laws of large numbers and optimal regret in online classification. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 447–455.

[5] Noga Alon, Yossi Matias, and Mario Szegedy. 1999. The Space Complexity of Approximating the Frequency Moments. *J. Comput. Syst. Sci.* 58, 1 (1999), 137–147.

[6] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. 2018. Synthesizing Robust Adversarial Examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML*. 284–293.

[7] Idan Attias, Edith Cohen, Moshe Shechner, and Uri Stemmer. 2021. A Framework for Adversarial Streaming via Differential Privacy and Difference Estimators. *CoRR* abs/2107.14527 (2021).

[8] Dmitrii Avdiukhin, Slobodan Mitrovic, Grigory Yaroslavtsev, and Samson Zhou. 2019. Adversarially Robust Submodular Maximization under Knapsack Constraints. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD*. 148–156.

[9] Mihir Bellare and Phillip Rogaway. 1993. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security*. 62–73.

[10] Mihir Bellare and Phillip Rogaway. 1996. The Exact Security of Digital Signatures - HOw to Sign with RSA and Rabin. In *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Proceeding*. 399–416.

[11] Omri Ben-Eliezer, Talya Eden, and Krzysztof Onak. 2022. Adversarially Robust Streaming via Dense-Sparse Trade-offs. In *5th Symposium on Simplicity in Algorithms, SOSA*. (to appear).

[12] Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. 2021. A Framework for Adversarially Robust Streaming Algorithms. *SIGMOD Rec.* 50, 1 (2021), 6–13.

[13] Omri Ben-Eliezer and Eylon Yogev. 2020. The Adversarial Robustness of Sampling. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS*. 49–62.

[14] Arnab Bhattacharyya, Palash Dey, and David P. Woodruff. 2019. An Optimal Algorithm for L_1-Heavy Hitters in Insertion Streams and Related Problems. *ACM Trans. Algorithms* 15, 1 (2019), 2:1–2:27.

[15] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion Attacks against Machine Learning at Test Time. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD, Proceedings, Part III*. 387–402.

[16] Ilija Bogunovic, Slobodan Mitrovic, Jonathan Scarlett, and Volkan Cevher. 2017. Robust Submodular Maximization: A Non-Uniform Partitioning Approach. In *Proceedings of the 34th International Conference on Machine Learning, ICML*. 508–516.

[17] Vladimir Braverman, Avinatan Hassidim, Yossi Matias, Mariano Schain, Sandeep Silwal, and Samson Zhou. 2021. Adversarial Robustness of Streaming Algorithms through Importance Sampling. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, NeurIPS*. (to appear).

[18] Paul Brown, Peter J. Haas, Jussi Myllymaki, Hamid Pirahesh, Berthold Reinwald, and Yannis Sismanis. 2005. Toward Automated Large-Scale Information Integration and Discovery. In *Data Management in a Connected World, Essays Dedicated to Hartmut Wedekind on the Occasion of His 70th Birthday*. 161–180.

[19] Harry Buhrman, Richard Cleve, and Avi Wigderson. 1998. Quantum vs. classical communication and computation. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. 63–68.

[20] Ran Canetti, Oded Goldreich, and Shai Halevi. 2004. The random oracle methodology, revisited. *Journal of the ACM (JACM)* 51, 4 (2004), 557–594.

[21] Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. 2022. Adversarially Robust Coloring for Graph Streams. In *13th Innovations in Theoretical Computer Science Conference, ITCS*. (to appear).

[22] Timothy M. Chan. 2010. A dynamic data structure for 3-D convex hulls and 2-D nearest neighbor queries. *J. ACM* 57, 3 (2010), 16:1–16:15.

[23] Timothy M. Chan and Qizheng He. 2021. More Dynamic Data Structures for Geometric Set Cover with Sublinear Update Time. In *37th International Symposium on Computational Geometry, SoCG*. 25:1–25:14.

[24] Yeshwanth Cherapanamjeri and Jelani Nelson. 2020. On Adaptive Distance Estimation. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS*.

[25] Graham Cormode, Mayur Datar, Piotr Indyk, and S. Muthukrishnan. 2003. Comparing Data Streams Using Hamming Norms (How to Zero In). *IEEE Trans. Knowl. Data Eng.* 15, 3 (2003), 529–540.

[26] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. 2018. AutoAugment: Learning Augmentation Policies from Data. *CoRR* abs/1805.09501 (2018).

[27] Tamraparni Dasu, Theodore Johnson, S. Muthukrishnan, and Vladislav Shkapenyuk. 2002. Mining database structure; or, how to build a data quality browser. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*. 240–251.

[28] David J. DeWitt, Jeffrey F. Naughton, Donovan A. Schneider, and S. Seshadri. 1992. Practical Skew Handling in Parallel Joins. In *18th International Conference on Very Large Data Bases, Proceedings*. Morgan Kaufmann, 27–40.

[29] James R. Driscoll, Neil Sarnak, Daniel Dominic Sleator, and Robert Endre Tarjan. 1989. Making Data Structures Persistent. *J. Comput. Syst. Sci.* 38, 1 (1989), 86–124.

[30] Cynthia Dwork, Moni Naor, Toniann Pitassi, Guy N. Rothblum, and Sergey Yekhanin. 2010. Pan-Private Streaming Algorithms. In *Innovations in Computer Science - ICS,Proceedings*, Andrew Chi-Chih Yao (Ed.). 66–80.

[31] Amos Fiat and Haim Kaplan. 2003. Making data structures confluently persistent. *J. Algorithms* 48, 1 (2003), 16–58.

[32] Anna C. Gilbert, Brett Hemenway, Martin J. Strauss, David P. Woodruff, and Mary Wootters. 2012. Reusable low-error compressive sampling schemes through privacy. In *IEEE Statistical Signal Processing Workshop, SSP*. 536–539.

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

[34] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and Harnessing Adversarial Examples. *CoRR* abs/1412.6572 (2014). arXiv:1412.6572 http://arxiv.org/abs/1412.6572

[35] André Gronemeier and Martin Sauerhoff. 2009. Applying Approximate Counting for Computing the Frequency Moments of Long Data Streams. *Theory Comput. Syst.* 44, 3 (2009), 332–348.

[36] Moritz Hardt and David P. Woodruff. 2013. How robust are linear sketches to adaptive inputs?. In *Symposium on Theory of Computing Conference, STOC*. 121–130.

[37] Avinatan Hassidim, Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. 2020. Adversarially Robust Streaming Algorithms via Differential Privacy. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*.

[38] Sandy H. Huang, Nicolas Papernot, Ian J. Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial Attacks on Neural Network Policies. In *5th International Conference on Learning Representations, ICLR*.

[39] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. 2018. Prior Convictions: Black-Box Adversarial Attacks with Bandits and Priors. *CoRR* abs/1807.07978 (2018).

[40] Rajesh Jayaram and David P. Woodruff. 2018. Data Streams with Bounded Deletions. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*. 341–354.

[41] Rajesh Jayaram and David P. Woodruff. 2019. Towards Optimal Moment Estimation in Streaming and Distributed Models. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*. 29:1–29:21.

[42] Haim Kaplan. 2004. Persistent Data Structures. In *Handbook of Data Structures and Applications*. Chapman and Hall/CRC.

[43] Haim Kaplan, Yishay Mansour, Kobbi Nissim, and Uri Stemmer. 2021. Separating Adaptive Streaming from Oblivious Streaming Using the Bounded Storage Model. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO, Proceedings, Part III*. 94–121.

[44] Jonathan Katz and Yehuda Lindell. 2014. *Introduction to Modern Cryptography, Second Edition*. CRC Press.

[45] Neal Koblitz and Alfred J Menezes. 2015. The random oracle model: a twenty-year retrospective. *Designs, Codes and Cryptography* 77, 2 (2015), 587–610.

[46] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *CoRR* abs/1607.02533 (2016). arXiv:1607.02533 http://arxiv.org/abs/1607.02533

[47] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2017. Adversarial Machine Learning at Scale. In *5th International Conference on Learning Representations, ICLR, Conference Track Proceedings*.

[48] Sam Lightstone. 2018. Physical Database Design for Relational Databases. In *Encyclopedia of Database Systems, Second Edition*. Springer.

[49] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2017. Delving into Transferable Adversarial Examples and Black-box Attacks. In *5th International Conference on Learning Representations, ICLR, Conference Track Proceedings*.

[50] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *6th International Conference on Learning Representations, ICLR,*

Conference Track Proceedings.
[51] Boaz Menuhin and Moni Naor. 2021. Keep That Card in Mind: Card Guessing with Limited Memory. *CoRR* abs/2107.03885 (2021).
[52] Daniele Micciancio and Chris Peikert. 2013. Hardness of SIS and LWE with Small Parameters. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference. Proceedings, Part I.* 21–39.
[53] Darakhshan J. Mir, S. Muthukrishnan, Aleksandar Nikolov, and Rebecca N. Wright. 2011. Pan-private algorithms via statistics on sketches. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS.* 37–48.
[54] Ilya Mironov, Moni Naor, and Gil Segev. 2011. Sketching in Adversarial Environments. *SIAM J. Comput.* 40, 6 (2011), 1845–1870.
[55] Jayadev Misra and David Gries. 1982. Finding Repeated Elements. *Sci. Comput. Program.* 2, 2 (1982), 143–152.
[56] Robert H. Morris Sr. 1978. Counting Large Numbers of Events in Small Registers. *Commun. ACM* 21, 10 (1978), 840–842.
[57] Moni Naor and Eylon Yogev. 2019. Bloom Filters in Adversarial Environments. *ACM Trans. Algorithms* 15, 3 (2019), 35:1–35:30.
[58] Jelani Nelson, Huy L. Nguyên, and David P. Woodruff. 2012. On Deterministic Sketching and Streaming for Sparse Recovery and Norm Estimation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM. Proceedings.* 627–638.
[59] Sriram Padmanabhan, Bishwaranjan Bhattacharjee, Timothy Malkemus, Leslie Cranston, and Matthew Huras. 2003. Multi-Dimensional Clustering: A New Data Layout Scheme in DB2. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data.* 637–641.
[60] Mohammad Roghani, Amin Saberi, and David Wajc. 2022. Beating the Folklore Algorithm for Dynamic Matching. In *13th Innovations in Theoretical Computer Science Conference, ITCS.* (to appear).
[61] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. 2018. Adversarially Robust Generalization Requires More Data. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS.* 5019–5031.
[62] Patricia G. Selinger, Morton M. Astrahan, Donald D. Chamberlin, Raymond A. Lorie, and Thomas G. Price. 1979. Access Path Selection in a Relational Database Management System. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data.* ACM, 23–34.
[63] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. 2016. Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security.* 1528–1540.
[64] Amit Shukla, Prasad Deshpande, Jeffrey F. Naughton, and Karthikeyan Ramasamy. 1996. Storage Estimation for Multidimensional Aggregates in the Presence of Hierarchies. In *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases.* 522–531.
[65] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. *International Conference on Learning Representations* (2014).
[66] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. 2018. Ensemble Adversarial Training: Attacks and Defenses. In *6th International Conference on Learning Representations, ICLR, Conference Track Proceedings.*
[67] Vinod Vaikuntanathan. 2015. Lecture Notes for Advanced Topics in Cryptography: Lattices.
[68] David Wajc. 2020. Rounding dynamic matchings against an adaptive adversary. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC.* 194–207.
[69] David P. Woodruff and Samson Zhou. 2021. Tight Bounds for Adversarially Robust Streams and Sliding Windows via Difference Estimators. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS.* 1183–1196.

A MISSING PROOFS

A.1 Missing Proofs from Section 2.2

Theorem 1.5. *Let $c \in (0, 1/2)$ and assume the adversary cannot solve the SIS problem of Definition 2.7 with parameter $\beta_\infty = \text{poly}(n)$ in Theorem 2.8 for sufficiently large n. Then Algorithm 3 returns a n^ε multiplicative approximation to the L_0 estimation problem. Furthermore, the algorithm uses space $n^{1-\varepsilon+c\varepsilon} + n^{(1+c)\varepsilon}$. In the random oracle model, the algorithm uses space $n^{1-\varepsilon+c\varepsilon}$.*

Proof. By Theorem 2.8, we know that the adversary cannot find any x such that $Ax = 0$ and $\|x\|_\infty \leq \text{poly}(n)$. Thus if the vectors we track using A equal 0, we know that none of the coordinates in that chunk have a positive f_i value at the end. Similarly if the vector is nonzero, we know that there is at least one (and at most n^ε) coordinates associated with that chunk that have nonzero frequency value at the end. Thus on each chunk of coordinates, we make multiplicative error at most n^ε and the theorem follows.

For the space bound, note that we can generate the appropriate column of A on the fly via access to the random oracle (or we can store A explicitly if we do not use the random oracle model). Thus the only space used is to keep track of the $n^{1-\varepsilon}$ vectors of size $n^{c\varepsilon}$, each associated with an n^ε chunk of coordinates of $[n]$. □

We also remark that Algorithm 2 can be used to estimate the inner product of two vectors f and g that are implicitly defined through two streams by using the following observations:

Lemma A.1. *[40] Let $f', g' \in \mathbb{R}^n$ be unscaled uniform samples of f and g, sampled with probability $p_f \geq \frac{s}{m_f}$ and $p_g \geq \frac{s}{m_g}$, where $s = \frac{1}{\varepsilon^2}$. Then with probability at least 0.99, we have*

$$\langle p_f^{-1} f', p_g^{-1} g' \rangle = \langle f, g \rangle \pm \varepsilon \|f\|_1 \|g\|_1.$$

Lemma A.2. *[58] Given $f, g \in \mathbb{R}^n$, suppose f' and g' are vectors that satisfy*

$$\|f' - f\|_\infty \leq \varepsilon \|f\|_1, \qquad \|g' - g\|_\infty \leq \varepsilon \|g\|_1.$$

Then $\langle f', g' \rangle - \langle f, g \rangle| \leq 12\varepsilon \|f\|_1 \|g\|_1$.

Combining Lemma A.1 and Lemma A.2, we have the following:

Corollary 2.6. *There exists a white-box adversarially robust algorithm that uses space $O\left(\frac{1}{\varepsilon}\left(\log n + \log \frac{1}{\varepsilon}\right) + \log\log m\right)$ and outputs vectors $f', g' \in \mathbb{R}^n$ such that with probability at least 3/4,*

$$|\langle f', g' \rangle - \langle f, g \rangle| \leq \varepsilon \|f\|_1 \|g\|_1.$$

A.2 Missing Proofs from Section 3.1

Theorem 1.8. *(Informal) Suppose there exists a white-box adversarially robust streaming algorithm using $S(n, \varepsilon)$ space that can be used to solve a one-way two-player communication game with $S(n, \varepsilon)$ bits of communication with probability $p \in (1/2, 1]$. Then there exists a deterministic protocol for the two-player communication game using $S(n, \varepsilon)$ bits of communication.*

Proof. Over all choices of randomness, at least p fraction of the possible random strings chosen by the first player is correct over all possible inputs to the second player and at least p fraction of the possible random strings chosen by the second player for each input. The first player can enumerate over all possible inputs to the second player as well as all possible random strings to select a state that uses $S(n, \varepsilon)$ space to represent and always succeeds. The first player can then send the state of the algorithm to the second player, who will then update the algorithm with their input and a fixed string, thereby resulting in a deterministic protocol that solves the one-way two-player communication game with $S(n, \varepsilon)$ bits of communication. □

We use the Gap Equality problem to show lower bounds for F_p moment estimation in the white-box adversarial streaming model:

Theorem 3.3. *For any $p \geq 0$ with $p \neq 1$, there exists a constant $C_p > 1$ such that any white-box adversarially robust algorithm that outputs an C_p-approximation to F_p with probability at least $9/10$ requires $\Omega(n)$ space.*

Proof. First note that given vectors $u, v \in \{0,1\}^n$ with $|u| = |v| = \frac{n}{2}$ and $\mathsf{HAM}(u,v) \geq \frac{n}{10}$, there exists a constant C_p such that $C_p\|2u\|_p \leq \|u+v\|_p$ for $p \in [0,1)$ and $C_p\|u+v\|_p \leq \|2u\|_p$ for $p > 1$. Assume for the sake of contradiction that there exists an algorithm $\mathcal{A}$ that uses $o\left(\frac{n}{2^k}\right)$ space and k hidden private bits and outputs a C_p-approximation to F_p with probability at least $9/10$ in the white-box adversarial model. Given an instance of $\textsc{DetGapEQ}_n$, Alice receives a string $x \in \{0,1\}^n$ with $|x| = \frac{n}{2}$ and Bob receives a string $y \in \{0,1\}^n$ with $|y| = \frac{n}{2}$. Alice creates a stream S that induces the frequency vector x.

Because Alice and Bob must solve $\textsc{DetGapEQ}_n$ deterministically, k random bits cannot be selected. Instead, Alice runs a separate instance of $\mathcal{A}$ on S for each of the 2^k possible realizations of the k random bits. For the i-th realization of the sequence of the hidden random bits under some fixed ordering, Alice deterministically chooses a sequence R_i of public random bits such that $\mathcal{A}$ is correct for at least $9/10$ fraction of the possible values of y. Otherwise, if such a sequence does not exist, Alice sets the sequence R_i to be the all zeros sequence. For each $i \in [2^k]$, Alice then runs the algorithm $\mathcal{A}$ on the i-th realization of the sequence of the hidden random bits under some fixed ordering, the deterministic fixing of the sequence R_i to use as $\mathcal{A}$'s public "random bits", and the input x to create a state $\sigma_i(x)$. Alice then sends the set of states $\sigma_1(x), \ldots, \sigma_{2^k}(x)$ to Bob.

Bob creates a stream that induces the frequency vector y. For each $i \in [2^k]$, Bob takes $\sigma_i(x)$, continues running $\mathcal{A}$ on the created stream, so that the underlying frequency vector is $x+y$, and queries the algorithm. Since a C_p-approximation to the norm of $x+y$ distinguishes whether $x = y$ or $\mathsf{HAM}(x,y) \geq \frac{n}{10}$ by the above argument, then for each $i \in [2^k]$, Bob can determine whether the i-th instance of $\mathcal{A}$ outputs whether $x = y$ or $\mathsf{HAM}(x,y) \geq \frac{n}{10}$ by enumerating over all possible "random" strings and taking the majority output by the algorithm. By the correctness of $\mathcal{A}$ in the white-box adversarial model, for at least $9/10$ of the possible realizations of the sequence of the hidden random bits under some fixed ordering will also be correct for *all possible values* of y, across at least $9/10$ of the possible public random bits used by the algorithm. Thus at least $9/10$ fraction of the states $\sigma_i(x)$ sent by Alice, where $i \in [2^k]$, will succeed for all possible values of y. Hence, at least $9/10$ fraction of the 2^k outputs by Bob will be correct, allowing Bob to distinguish whether $x = y$ or $\mathsf{HAM}(x,y) \geq \frac{n}{10}$.

From our assumption, each instance of $\mathcal{A}$ uses $o\left(\frac{n}{2^k}\right)$ space. Thus the states $\sigma_i(x)$ sent by Alice, where $i \in [2^k]$, use at most $o(n)$ communication, which contradicts Theorem 3.2. It follows that $\mathcal{A}$ uses $\Omega\left(\frac{n}{2^k}\right)$ space. □

A.3 Missing Proofs from Section 3.2

We now prove Theorem 1.11. Assume that there is an algorithm which, using s bits of memory, counts the number of 1's in a stream, consisting of 0's and 1's, approximately. A key result of this section is that to count up to n 1's up to a factor of $1+\varepsilon$, where $\varepsilon > 0$ is constant, we must have $s = \Omega(\log n)$; this asymptotically matches the bound required to count it exactly. Moreover the bound that holds even if the streaming algorithm has access to a clock that keeps track of the index of the input being read (the algorithm is not charged for the memory required to store the index). Such an algorithm can be modeled as an oblivious leveled read-once branching program (also known as Ordered Binary Decision Diagram, abbreviated as OBDD) of width 2^s over the input alphabet $\{0,1\}$. (Without loss of generality, we let the stream be infinite and therefore the length of the OBDD is also infinite.)

More generally, we show that the lower bound applies to a larger class of counting functions called *monotonic counters*.

Definition A.3 (Monotonic Counters)**.** *Let Σ be the input alphabet. A monotonic counter is a function $\chi : \Sigma^* \to \mathbb{N}\backslash\{0\}$ satisfying $\chi(\epsilon) = 1$ for the empty string[1] and $\{\chi(\sigma a) - \chi(\sigma) : a \in \Sigma\} = \{0,1\}$, for every $\sigma \in \Sigma^*$.*

In words, the counter is initialized to 1 at time 1. The counter can increase by at most 1 in each time step, and stays the same for at least one input symbol and strictly increases by 1 for at least one input symbol, thereby ensuring that the counter can assume all possible values in $\{1, 2, \ldots, t\}$ at time t.

Fix a monotonic counter χ over an input alphabet Σ. Let P be an OBDD also over Σ and let $P(\sigma)$ denote the node reached in P on input $\sigma \in \Sigma^*$. Fix any node u in P and let $C_u = \{\chi(\sigma) : P(\sigma) = u \text{ for some } \sigma\}$ be the nonempty set of values of the monotonic counter for input sequences that reach node u.

To characterize the error in P's computation, we abstractly let $\varepsilon : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be a function that represents the error in approximation. For example:

(1) $\varepsilon(k) = \delta k$ where $\delta > 0$ is a fixed constant $\implies$ $(1+\delta)$-multiplicative approximation.
(2) $\varepsilon(k) = (n^\delta - 1)k$ where $0 < \delta < 1$ is a fixed constant $\implies$ n^δ-multiplicative approximation.
(3) $\varepsilon(k) = n^\delta$ where $0 \leq \delta < 1$ is a fixed constant $\implies$ n^δ-additive approximation.

We say that a set C of values is *ε-bound* if the deviation of its maximum value from k is at most $\varepsilon(k)$ for every $k \in C$. We say that P has error ε at time t if C_u is ε-bound for every node u at time t.

For a node u, let $J_u = [\min(C_u), \max(C_u)]$ be the interval that minimally covers C_u, and let $I_0(t)$ be the set of intervals J_u over all nodes u at level t. Define $I(t)$ to be the set of all maximal intervals (under set inclusion) in $I_0(t)$. Note that $|I(t)|$ is a lower bound on the number of nodes in P at time t. For this section, it suffices to consider the error in approximation induced by the intervals in $I(t)$. Namely, if P has error ε at time t, then it implies every interval of $I(t)$ is ε-bound.

Lemma A.4. *$I(1) = \{[1,1]\}$.*

Proof. This holds because the initial value of the monotonic counter is 1. □

Lemma A.5. *Let $t' \geq t \geq 1$. For every interval $[k,\ell] \in I(t)$, there exists an interval in $I(t')$ containing $[k,\ell]$.*

[1] This does not entail a loss of generality for monotonic counters that can also output 0; such exceptional sequences can be handled separately.

Proof. The statement is trivially true for $t' = t$ and we show that it holds for $t' = t+1$; it follows by induction that also holds for all $t' \geq t$. Let u be the node in level t such that $J = J_u = [k, \ell]$ is the interval associated with node u. Let σ be an input sequence such that $P(\sigma) = u$ and $\chi(\sigma) = k$. Similarly, let ρ be another input sequence such that $P(\rho) = u$ and $\chi(\rho) = \ell$.

Let $a \in \Sigma$ be such that $\chi(\sigma a) = \chi(\sigma) = k$. Let $v = P(\sigma a)$ be the node reached in level $t+1$. Then $v = P(\rho a)$ as well and $\chi(\rho a) \geq \chi(\rho) = \ell$. Thus $J_v \supseteq [k, \ell]$ implying that there is an interval in $I(t+1)$ that contains $[k, \ell]$. □

Lemma A.6. *Let $t \geq 1$. For every interval $[k, \ell] \in I(t)$, there exists an interval in $I(t+1)$ containing $[k+1, \ell+1]$.*

Proof. Let u, σ and ρ be as in the proof of Lemma A.5. Let $a \in \Sigma$ be such that $\chi(\rho a) = \chi(\rho) + 1 = \ell + 1$. Let $v = P(\rho a)$ be the node reached in level $t+1$. Then $v = P(\sigma a)$ as well and $\chi(\sigma a) \leq \chi(\sigma) + 1 = k + 1$. Thus $J_v \supseteq [k+1, \ell+1]$ implying that there is an interval in $I(t+1)$ that contains $[k+1, \ell+1]$. □

Now fix any family $\{I(t)\}_{t \geq 1}$, where $I(t)$ for each "time" t is a set of maximal intervals. The proof below depends only on the assumption the family satisfies Lemma A.4, Lemma A.5, and Lemma A.6 above. Our goal is to give a lower bound on $|I(t)|$ for some appropriate time t. We say that k is *present* at time t if there exists an interval in $I(t)$ whose left endpoint is k.

Lemma A.7. *1 is present at all times.*

Proof. By Lemma A.4, $[1, 1] \in I(1)$, and by Lemma A.5, $I(t)$ contains an interval $[k, \ell] \supseteq [1, 1]$ for each $t \geq 1$. Because $k \geq 1$ we have $k = 1$. □

We say that k is *exceptional* at time t if k is present at time t but $k+1$ is absent at time $t+1$. We show first that if exceptional counts occur very few times then we get a lower bound $|I(t)|$ for some time t. Fix a time horizon n and the corresponding subfamily $\{I(t)\}_{t=1}^{n+1}$. For each $h \geq 1$, let ϕ_h be the number of times $t \in \{1, 2, \ldots, n\}$ that some count $1 \leq k \leq h$ is exceptional at time t.

Lemma A.8. *If $(\phi_h + 1)h \leq n$, then there exists $t_0 \in \{1, 2, \ldots, n+1\}$ such that $|I(t_0)| \geq h + 1$.*

Proof. In the interval $[1, n]$, mark the times t when some count $1 \leq k \leq h$ is exceptional at time t. The unmarked times can be represented as a disjoint union of $\phi_h + 1$ intervals. By the pigeonhole principle, there exists one interval with size at least h. Let $\{t, t+1, \ldots, t+h-1\}$ be such that no count $1 \leq k \leq h$ is exceptional in that interval. We show by induction that every count in $\{1, 2, \ldots, k\}$ is present at time $t + k - 1$ for $1 \leq k \leq h+1$. That would imply for $k = h+1$ that $|I(t+h)| \geq h+1$. Since $t + h - 1 \leq n$, the lemma holds with $t_0 = t + h$.

The base case $k = 1$ follows by Lemma A.7. Assume the statement holds for some $1 \leq k \leq h$. Then, since each count in $\{1, 2, \ldots, k\}$ is non-exceptional at time $t + k - 1$, it follows by definition that each count in $\{2, \ldots, k+1\}$ is present at time $t + k$. Together with Lemma A.7, it follows that each count in $\{1, 2, \ldots, k+1\}$ is present at time $t + k$ as well. □

We will now bound ϕ_h so that the above lemma can be applied for h as large as possible. The idea below is to show that if a single count is exceptional too many times, then it belongs to some interval of large length, violating the approximability guarantee.

Lemma A.9. *Suppose k is exceptional at each time $t \in E$ for some nonempty set E. Then for all $t > \max(E)$, there exists an interval in $I(t)$ containing k whose right endpoint is at least $k + |E|$.*

Proof. For every $s \in E$, since k is exceptional at time s, it is also present at time s by definition. Let t_0 be the minimum value in E and let $[k, \ell_0] \in I(t_0)$ for some ℓ_0 certify k's presence at time t_0. We show the following statement, denoted $P(E, t_0, \ell_0)$, by induction on $|E|$: for each $t > \max(E)$ there exists an interval in $I(t)$ containing k such that its right endpoint is at least $\ell_0 + |E|$. Since $\ell_0 \geq k$, this proves the lemma.

For $|E| = 1$, because $[k, \ell_0] \in I(t_0)$, by Lemma A.6, there exists an interval $[k', \ell'] \supseteq [k+1, \ell_0 + 1]$ in $I(t_0 + 1)$. Because $k+1$ is absent at time $t_0 + 1$, we have $k' \leq k$. By Lemma A.5, there exists $J \in I(t)$ with $J \supseteq [k', \ell'] \supseteq [k, \ell_0 + 1]$. Thus, $k \in J$ and the right endpoint of J is at least $\ell_0 + 1 = \ell_0 + |E|$.

For $|E| > 1$, let $t_1 \neq t_0$ denote $\max(E)$. Let $[k, \ell_1] \in I(t_1)$ for some ℓ_1 certify k's presence at time t_1. Let $F = E \backslash \{t_1\}$ and observe that $t_0 = \min(F)$ as well. Apply the induction hypothesis $P(F, t_0, \ell_0)$: since $t_1 > \max(F)$, there exists an interval in $I(t_1)$ containing k such that its right endpoint is at least $\ell_0 + |F|$. On the other hand, $[k, \ell_1]$ is maximal in $I(t)$, therefore $\ell_1 \geq \ell_0 + |F|$.

Apply the induction hypothesis $P(\{t_1\}, t_1, \ell_1)$: for every $t > t_1 = \max(E)$, there exists an interval in $I(t)$ containing k such that its right endpoint is at least $\ell_1 + 1 \geq \ell_0 + |F| + 1 = \ell_0 + |E|$. □

Finally, we consider the situation where every element of $I(n+1)$ is ε-bound. By Lemma A.9, each k can be exceptional at most $\varepsilon(k)$ times in $\{1, 2, \ldots, n\}$. This gives an immediate bound on ϕ_h namely $\phi_h \leq \sum_{k=1}^{h} \varepsilon(k)$. To apply Lemma A.8, we find the largest h such that $\left(1 + \sum_{k=1}^{h} \varepsilon(k)\right) h \leq n$.

For example, when $\varepsilon(k) = \delta k$, where $\delta > 0$ is constant, we have $\sum_{k=1}^{h} \varepsilon(k) \leq \delta h(h+1)/2$. Therefore there exists a good choice of h with $h = \Theta(n^{1/3})$ that yields a polynomial bound for $|I(t_0)|$. For a multiplicative approximation $n^{1-\delta}$, where $\delta > 0$ is constant, we can choose $h = \Theta(n^{\delta'/3})$ for some $0 < \delta' < \delta$ and still obtain a polynomial bound for $|I(t_0)|$. Finally, for an additive approximation $n^{1-\delta}$, we can choose $h = \Theta(n^{\delta'/2})$ for some $0 < \delta' < \delta$.

The proof of Theorem 1.11 follows by noting that since $|I(t_0)| = \Omega(\text{poly}(n))$, the number of states in any deterministic approximate counting algorithm must also be $\Omega(\text{poly}(n))$. Therefore, the algorithm requires at least $\Omega(\log n)$ bits of space.