Improved Algorithms for White-Box Adversarial Streams

Ying Feng [1] David P. Woodruff [1]

Abstract

We study streaming algorithms in the white-box adversarial stream model, where the internal state of the streaming algorithm is revealed to an adversary who adaptively generates the stream updates, but the algorithm obtains fresh randomness unknown to the adversary at each time step. We incorporate cryptographic assumptions to construct robust algorithms against such adversaries. We propose efficient algorithms for sparse recovery of vectors, low rank recovery of matrices and tensors, as well as low rank plus sparse recovery of matrices, i.e., robust PCA. Unlike deterministic algorithms, our algorithms can report when the input is not sparse or low rank even in the presence of such an adversary. We use these recovery algorithms to improve upon and solve new problems in numerical linear algebra and combinatorial optimization on white-box adversarial streams. For example, we give the first efficient algorithm for outputting a matching in a graph with insertions and deletions to its edges provided the matching size is small, and otherwise we declare the matching size is large. We also improve the approximation versus memory tradeoff of previous work for estimating the number of non-zero elements in a vector and computing the matrix rank.

1. Introduction

The streaming model captures key resource requirements of algorithms for database, machine learning, and network tasks, where the size of the data is significantly larger than the available storage, such as for internet and network traffic, financial transactions, simulation data, and so on. This model was formalized in the work of Alon, Matias, and Szegedy (Alon et al., 1996), which models a vector undergoing additive updates to its coordinates. Formally, there is an underlying n-dimensional vector x, which could be a flattened matrix or tensor, which is initialized to 0^n and evolves via an arbitrary sequence of $m \leq \operatorname{poly}(n)$ additive updates to its coordinates. These updates are fed into a streaming algorithm, and the t-th update has the form (i_t, δ_t), meaning $x_{i_t} \leftarrow x_{i_t} + \delta_t$. Here $i_t \in \{1, 2, \ldots, n\}$ and $\delta_t \in \{-M, -M+1, \ldots, M-1, M\}$ for an $M \leq \operatorname{poly}(n)$[1]. Throughout the stream, x is promised to be in $\{-M, -M+1, \ldots, M-1, M\}^n$. A streaming algorithm makes one pass over the stream of updates and uses limited memory to approximate a function of x.

A large body of work on streaming algorithms has been designed for *oblivious* streams, for which the sequence of updates may be chosen adversarially, but it is chosen independently of the randomness of the streaming algorithm. In practical scenarios, this assumption may not be reasonable; indeed, even if the stream is not generated by an adversary this may be problematic. For example, if one is running an optimization procedure, then one may feed future data into a streaming algorithm based on past outputs of that algorithm, at which point the inputs depend on the algorithm's randomness and there is no guarantee of correctness. This is also true in recommendation systems, where a user may choose to remove suggestions based on previous queries.

There is a growing body of work on streaming algorithms that are robust in the black-box adversarial streaming model (Ben-Eliezer & Yogev, 2020; Ben-Eliezer et al., 2021; Hassidim et al., 2020; Woodruff & Zhou, 2021; Alon et al., 2021; Kaplan et al., 2021; Braverman et al., 2021; Menuhin & Naor, 2021; Attias et al., 2021; Ben-Eliezer et al., 2022; Chakrabarti et al., 2022), in which the adversary can monitor only the output of the streaming algorithm and choose future stream updates based on these outputs. While useful in a number of applications, there are other settings where the adversary may also have access to the internal state of the algorithm, and this necessitates looking at a stronger adversarial model known as white-box adversarial streams.

[1] Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA. Correspondence to: Ying Feng <yingfeng@andrew.cmu.edu>, David P. Woodruff <dwoodruf@cs.cmu.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[1] All bounds can be generalized to larger m and M; this is only for convenience of notation. Also, our streaming model, which allows for both positive and negative updates, is referred to as the (standard) turnstile streaming model in the literature.

1.1. The White-box Adversarial Streaming Model

We consider the white-box adversarial streaming model, introduced in (Ajtai et al., 2022), where a sequence of stream 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.

Definition 1.1. *(White-box Adversarial Streaming Model) Consider a single-pass, two-player game between Streamalg, the streaming algorithm, and Adversary.*

Prior to the beginning of the game, fix a query $\mathcal{Q}$, which asks for a function of an underlying dataset that will be implicitly defined by the stream, which is itself chosen by Adversary. The game then proceeds across m rounds, where in the t-th round:

1. Adversary computes an update u_t for the stream, which depends on all previous stream updates, all previous internal states, and randomness used by Streamalg (and thus also, all previous outputs of Streamalg).

2. Streamalg acquires a fresh batch R_t of random bits, uses u_t and R_t to update its data structures D_t, and (if asked) outputs a response A_t to the query $\mathcal{Q}$.

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 $\mathcal{Q}$ at some time $t \in [m]$ throughout the stream.

Notation: A function $f(n)$ is said to be *negligible* if for every polynomial $P(n)$, for all large enough n, $f(n) < \frac{1}{P(n)}$. We typically denote negligible functions by $\text{negl}(n)$.

Given a fixed time bound $\mathcal{T}$, we say a streaming algorithm is robust against $\mathcal{T}$ time-bounded white-box adversaries if no $\mathcal{T}$ time-bounded white-box adversary can win the game with non-negligible probability against this algorithm.

1.1.1. Applications of white-box adversaries

The white-box adversarial model captures characteristics of many real-world attacks, where an adaptive adversary has access to the entirety of the internal states of the system. In comparison to the oblivious stream model or the black-box adversarial model (Ben-Eliezer et al., 2021), this model allows us to model much richer adversarial scenarios.

For example, consider a distributed streaming setting where a centralized server collects statistics of a database generated by remote users. The server may send components of its internal state S to the remote users in order to optimize the total communication over the network. The remote users may use S in some process that generates downstream data. Thus, future inputs depend on the internal state S of the streaming algorithm run by the central coordinator. In such settings, the white-box robustness of the algorithms is crucial for optimal selection of query plans (Selinger et al., 1979), online analytical processing (Shukla et al., 1996; Padmanabhan et al., 2003), data integration (Brown et al., 2005), and data warehousing (Dasu et al., 2002).

Many persistent data structures provide the ability to quickly access previous versions of information stored in a repository shared across multiple collaborators. 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 previous states (Driscoll et al., 1989; Fiat & Kaplan, 2003; Kaplan, 2004).

Dynamic algorithms often consider an adaptive adversary who generates the updates upon seeing the entire data structure maintained by the algorithm during the execution (Chan, 2010; Chan & He, 2021; Roghani et al., 2022). For example, (Wajc, 2020) assumes the entire state of the algorithm (including the set of randomness) is available to the adversary after each update, i.e., a white-box model.

Moreover, robust algorithms and adversarial attacks are important topics in machine learning (Szegedy et al., 2014; Goodfellow et al., 2014), with a large body of recent literature focusing on adversarial robustness of machine learning models against white-box attacks (Ilyas et al., 2018; Madry et al., 2018; Schmidt et al., 2018; Tramèr et al., 2018; Cubuk et al., 2018; Kurakin et al., 2017; Liu et al., 2017). There exist successful attacks that use knowledge of the trained model; e.g., the weights of a linear classifier to minimize the loss function (which are referred to as Perfect Knowledge adversaries in (Biggio et al., 2013)). There are also white-box attacks that use the architecture and parameters of a trained neural network policy to generate adversarial perturbations that are almost imperceptible to the human eye but result in misclassification by the network (Huang et al., 2017). In comparison to the black-box adversarial streaming model, in which the input is chosen by an adversary who repeatedly queries for only a fixed property of the underlying dataset at each time but does not see the full internal state of the algorithm during execution, the white-box model more effectively captures the full capability of these attacks.

1.2. Random Oracle Model

In order to construct streaming algorithms based on the hardness of the SIS problem, we need access to a fixed uniformly random matrix during the stream. In this paper, we consider algorithms in the *random oracle model*, which means that the algorithms, as well as the white-box adversaries, are given read access to an arbitrarily long string of random bits. Each query gives a uniformly 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 (Bellare & Rogaway, 1993; 1996; Canetti et al., 2004; Koblitz & Menezes, 2015). Also, such a model has been used to design space-efficient streaming algorithms, for both oblivious streams (Clifford & Cosma, 2013; Jayaram & Woodruff, 2023) as well as adversarial settings (Ajtai et al., 2022; Ben-Eliezer et al., 2020). In the random oracle model, instead of storing large random sketching matrices during the stream, the streaming algorithms can generate the columns of the matrix on the fly when processing updates. Also, in the distributed setting, the servers will be able to agree on a random sketching matrix without having to communicate it.

Such an oracle is often implemented with hash-based heuristic functions such as AES or SHA256. These implementations are appealing since they behave, as far as we can tell in practice, like random functions. They are also extremely fast and incur no memory cost.

Another approach is to use a *pseudorandom function* as a surrogate.

Definition 1.2. *(Pseudorandom Function) Let A, B be finite sets, and let $\mathcal{F} = \{F_i : A \to B\}$ be a function family, endowed with an efficiently sampleable distribution. We say that $\mathcal{F}$ is a pseudorandom function (PRF) family if all functions F_i are efficiently computable and the following two games are computationally indistinguishable:*

1. Sample a function $F \overset{\$}{\leftarrow} \mathcal{F}$ and give the adversary adaptive oracle access to $F(\cdot)$.

2. Choose a uniformly random function $U : A \to B$ and give the adversary adaptive oracle access to $U(\cdot)$.

Given a random key to draw F from $\mathcal{F}$, a pseudorandom function provides direct access to a deterministic sequence of pseudorandom bits. This pseudorandom bit sequence can be seen as indexed by indices in A. Moreover, the key size can be logarithmically small with respect to the function domain (though in our algorithms we only need the key size to be polynomially small).

In this work, we design algorithms based on hardness assumptions of lattice cryptographic problems. In particular, we use the Short Integer Solution (SIS) Problem; see Section 2 for the precise cryptographic assumptions we make. There are many existing schemes to construct families of pseudorandom functions based on cryptographic assumptions (Goldreich et al., 1986; Banerjee et al., 2011; Kim, 2021). Therefore, if we assume the hardness of the SIS problem against $\mathcal{T}$ time-bounded adversaries, then we can construct families of pseudorandom functions. Moreover, if a function $F \overset{\$}{\leftarrow} \mathcal{F}$ is sampled privately from any of these families $\mathcal{F}$, then F behaves just like a random oracle from the perspective of any $\mathcal{T}$ time-bounded adversary.

However, in the white-box adversarial setting, the process of choosing $F \overset{\$}{\leftarrow} \mathcal{F}$ is revealed to the adversary. So the adversary can distinguish cases 1 and 2 in Definition 1.2 by simply comparing the output with F. It may be possible to use a pseudorandom function in place of a random oracle in our algorithms if one can resolve the following question:

Let $\mathcal{F}$ be a family of pseudorandom functions, constructed based on the SIS problem. Consider a one-round, two player game between $\underline{\text{Challenger}}$ and $\underline{\text{Adversary}}$:

1. $\underline{\text{Challenger}}$ samples a pseudorandom function $F \overset{\$}{\leftarrow} \mathcal{F}$ based on some random key $\mathcal{K}$, and reveals $\mathcal{K}$ to $\underline{\text{Adversary}}$.

2. $\underline{\text{Challenger}}$ uses the pseudorandom bits generated by F to sample an instance $\mathcal{I}$ of the SIS problem, with hardness parameter $n = |\mathcal{K}|$.

3. $\underline{\text{Adversary}}$ attempts to solve $\mathcal{I}$.

Assuming that no $\mathcal{T}$ time-bounded adversary can solve the SIS problem with non-negligible probability, does there exist a $\mathcal{T}$ time-bounded adversary that can win this game with non-negligible probability, for a fixed time bound $\mathcal{T}$?

The answer to this question depends on the specific PRF construction that we use. One may be able to artificially construct a family of SIS-based PRFs and show that the pseudorandomness generated by such PRF induces an easy variant of the SIS problem. However, using other PRF constructions, the SIS problem could potentially retain its difficulty. We leave the question of removing our random oracle assumption as an interesting direction for future work.

1.3. Our Contributions

Table 1 summarizes our contributions. Specifically, we construct sparse recovery schemes for vectors, low rank plus sparse recovery schemes for matrices, and low rank recovery schemes for tensors, and apply these as building blocks to solve a number of problems in the white-box adversarial streaming model. Our algorithms either improve the bounds of existing algorithms, often optimally, or solve a problem for which previously no known white-box adversarial streaming algorithm was known.

1.3.1. Recovery Algorithms

We start by giving recovery algorithms for k-sparse vectors and rank-k matrices in the white-box adversarial streaming model, which reconstruct their input provided that it satisfies the sparsity or rank constraint. Our algorithms crucially have the property that *they can detect if their input violates the sparsity or rank constraint*.

Our algorithms make use of hardness assumptions of the Short Integer Solution (SIS) Problem and hold against polynomial (and sometimes larger) time adversaries. See Section 2 for the precise cryptographic assumptions we make. Informally, we have Theorem 1.3, Theorem 1.4, Theorem 1.5, and Theorem 1.6 below.

Theorem 1.3. *Assuming the exponential hardness of the SIS problem, there exists a white-box adversarially robust streaming algorithm which determines if the input vector is k-sparse, for parameter $k \geq n^c$ for an arbitrarily small constant $c > 0$, and if so, recovers a k-sparse vector using $\tilde{\mathcal{O}}(k)$ bits*[2] *of space in the random oracle model.*

We note that there are standard deterministic, and hence also white-box adversarially robust, k-sparse vector recovery schemes based on any deterministic algorithm for compressed sensing (Candès & Wakin, 2008). However, previous algorithms require the promise that the input is k-sparse; otherwise, their output can be arbitrary. That is, there is no way to know if the input is k-sparse or not. In contrast, our algorithm does not assume sparsity of the input, and reports a failure when the input is not k-sparse. We stress that this is not an artifact of analyses of previous algorithms; in fact, any deterministic streaming algorithm cannot detect if its input vector is k-sparse without using $\Omega(n)$ bits of memory (Ganguly & Majumder, 2006). By Theorem 2 in (Ajtai et al., 2022), this implies an $\Omega(n)$ bit lower bound for *any randomized* k-sparse decision algorithms in the white-box streaming model. Thus our algorithm provides a provable separation between computationally bounded and unbounded adversaries, under cryptographic assumptions.

While sparsity is a common way of capturing vectors described with few parameters, low rank is a common way of capturing matrices described with few parameters. We next extend our results to the matrix setting:

Theorem 1.4. *Assuming the exponential hardness of the SIS problem, there exists a white-box adversarially robust streaming algorithm which decides if an $n \times n$ input matrix with integer entries bounded by a polynomial in n, has rank at most k, and if so, recovers the matrix using $\tilde{\mathcal{O}}(nk)$ bits of space in the random oracle model* [3].

Theorem 1.4 provides the first low rank matrix recovery algorithm in the white-box streaming model. Moreover, the space complexity of this algorithm is nearly optimal, as just describing such a matrix requires $\Omega(nk \log n)$ bits. This result again provides a separation from deterministic algorithms under cryptographic assumptions, as a simple reduction from the Equality communication problem (see, e.g., (Alon et al., 1999) for similar reductions) shows that testing if the input matrix in a stream is all zeros or not requires $\Omega(n^2)$ memory.

In addition, our results can be further extended to recover a sparse plus low rank matrix for robust principal component analysis (robust PCA) (Chandrasekaran et al., 2011; Candès et al., 2011) and also can recover a low rank tensor:

Theorem 1.5. *Under Assumption 2.3, given parameters $r, k > 0$, there exists a streaming algorithm robust against $o(n^{nk+r})$ time-bounded white-box adversaries that determines if an $n \times n$ input matrix can be decomposed into the sum of a matrix with rank at most k and a matrix with at most r non-zero entries, and if so, finds the decomposition using $\tilde{\mathcal{O}}(nk + r)$ bits of space and $\mathrm{poly}(n)$ time in the random oracle model.*

Theorem 1.6. *For an input tensor $X \in \mathbb{Z}_q^{n_1 \times \cdots \times n_d}$ with $q \in \mathrm{poly}(n)$ for $n = \prod_1^d n_i$, under Assumption 2.3, given parameter k with $k \in \Theta(\frac{n^c}{(n_1 + \cdots + n_d) \log n})$ for a constant $c > 0$, there exists a streaming algorithm robust against $o(n^{k(n_1 + \cdots + n_d)})$ time-bounded white-box adversaries that determines if the input tensor has CP rank at most k and if so, recovers the tensor using $\tilde{\mathcal{O}}(k(n_1 + \cdots n_d))$ bits of space and $\mathrm{poly}(n)$ time in the random oracle model.*

See Appendices A and B for more on Theorems 1.5 and 1.6.

1.3.2. Applications

Our sparse recovery theorem for vectors can be used to simplify and improve the existing upper bound for the ℓ_0-norm estimation problem in the white-box adversarial model, which is the problem of estimating the support size, i.e., the number of non-zero entries of the input vector x.

Theorem 1.7. *(Informal) Assuming the exponential hardness of the SIS problem, there exists a white-box adversarially robust streaming algorithm which estimates the ℓ_0 norm within a factor of n^{ε} using $\tilde{\mathcal{O}}(n^{1-\varepsilon})$ bits of space in the random oracle model.*

Previously, the only known white-box adversarially robust algorithm for ℓ_0 norm estimation required $\tilde{\mathcal{O}}(n^{1-\varepsilon+c\varepsilon})$ space for an n^{ε}-approximation, where $c > 0$ is a fixed constant, in the random oracle model. Our algorithm replaces c with 0.

Based on our low rank matrix recovery algorithm, we give the first algorithm for finding a maximum matching in a graph if the maximum matching size is small, or declare that the maximum matching size is large, in a stream with insertions and deletions to its edges. Standard methods based on filling in the so-called Tutte matrix of a graph randomly do not immediately work, since the adversary sees this randomness in the white box model. Nevertheless, we show that filling in the Tutte matrix deterministically during the stream suffices for our purposes.

[2] Here and throughout, $\tilde{\mathcal{O}}(f)$ denotes $f \cdot \mathrm{poly}(\log n)$, with n defined in our description of the streaming model.

[3] Our results all generalize to $n \times d$ matrices; we state them here for square matrices for convenience only.

Theorem 1.8. *(Informal) Assuming the exponential hardness of the SIS problem, there is a white-box adversarially robust streaming algorithm using $\tilde{O}(nk)$ space in the random oracle model and $\mathrm{poly}(n)$ running time, which either declares the maximum matching size is larger than k, or outputs a maximum matching.*

We note that for any matrix problem, such as linear matroid intersection or parity or union, matrix multiplication and decomposition, finding a basis of the null space, and so on, if the input consists of low rank matrices then we can first recover the low rank matrix in a white-box adversarial stream, verify the input is indeed of low rank, and then run an offline algorithm for the problem, such as those in (Cheung et al., 2013).

Besides solving new problems, as an immediate corollary we also obtain an improved quantitative bound for testing if the rank of an input matrix is at most k, which is the rank decision problem of (Ajtai et al., 2022).

Theorem 1.9. *(Informal) Assuming the exponential hardness of the SIS problem, there exists a white-box adversarially robust streaming algorithm which solves the rank decision problem using $\tilde{\mathcal{O}}(nk)$ bits of space in the random oracle model.*

In (Ajtai et al., 2022), a weaker $\mathcal{O}(nk^2)$ space bound was shown for white-box adversarially robust algorithms. Our improvement comes by observing that we can get by with many fewer than k rows in our sketch, provided that the modulus q in the SIS problem (see Section 2) is large enough. This may be counterintuitive as the rank of our sketch may be much less than k but we can still recover rank-k inputs by using a large enough modulus to encode them.

2. Preliminaries

2.1. Short Integer Solution Problem

We make use of well-studied cryptographic assumptions in the design of our algorithms. Specifically, we construct white-box adversarially robust algorithms based on the assumed hardness of the Short Integer Solution problem.

Definition 2.1. *(Short Integer Solution (SIS) Problem) Let n, m, q be integers and let $\beta > 0$. Given a uniformly random matrix $A \in \mathbb{Z}_q^{n \times m}$ with $m \in \mathrm{poly}(n)$, the SIS problem is to find a non-zero integer vector $z \in \mathbb{Z}^m$ such that $Az = 0 \mod q$ and $\|z\|_2 \leq \beta$.*

Theorem 2.2. *(Micciancio & Peikert, 2013) Let n and $m, \beta, q \in \mathrm{poly}(n)$ be integers and $q \geq n \cdot \beta$. Then solving the SIS problem with non-negligible probability, with parameters n, m, q, β is at least as hard as γ-approximation of the Shortest Vector Problem (SVP$_\gamma$) with $\gamma \in \mathrm{poly}(n)$.*

Theorem 2.2 bases the hardness of the SIS problem on the SVP$_\gamma$ problem, which is one of the most well-studied lattice problems with many proposed algorithms. The best known algorithm for SVP$_\gamma$ with $\gamma = \mathrm{poly}(n)$ is due to (Aggarwal et al., 2015) and runs in $\tilde{\mathcal{O}}(2^n)$ time.

2.2. SIS Hardness Assumption

We assume that the white-box adversary is computationally bounded in such a way that it cannot solve the SIS problem with non-negligible probability. For the purposes of this paper, we consider a time bound based on the state-of-the-art complexity result for lattice problems.

Assumption 2.3. *Given $n \in \mathbb{N}$, for some $m, \beta, q \in \mathrm{poly}(n)$ and $q \geq n \cdot \beta$, no $o(2^n)$ time-bounded adversary can solve the SIS problem $\mathsf{SIS}_{n,m,p,\beta}$ with non-negligible probability.*

As shown above, our instance of the SIS problem is at least as hard as the approximation problem SVP$_{\mathrm{poly}(n)}$, for which the best-known algorithm runs in $\tilde{\mathcal{O}}(2^n)$ time.

We have the following two crucial lemmas.

Lemma 2.4. *Under Assumption 2.3, given a uniformly random matrix $A \in \mathbb{Z}_q^{n \times m}$ for $q, m, \beta \in \mathrm{poly}(n)$ and $q \geq n \cdot \beta$, if a vector $x \in \mathbb{Z}_\beta^m$ is generated by an $o(2^n)$-time adversary, then with probability at least $1 - negl(n)$, there does not exist a k-sparse vector $y \in \mathbb{Z}_\beta^m$ for which $x \neq y \mod q$ yet $Ax = Ay \mod q$, for $k \in o(\frac{n}{\log n})$.*

Remark 2.5. *We note that given a random matrix $A \in \mathbb{Z}_q^{n \times m}$, when both x and y are k-sparse, we can argue information-theoretically by a union bound that with high probability all sparse $x \neq y$ with bounded entries satisfy $Ax \neq Ay$. However, there may exist a binary vector x which is not k-sparse, and a k-sparse y with bounded entries such that $Ax = Ay$. In this case we need the SIS assumption to show that it is hard for an adversary to find such x and fool the algorithm.*

Proof. If an adversary were to find a vector $y \in \mathbb{Z}_\beta^m$ for which $x \neq y \mod q$ yet $Ax = Ay \mod q$, then it would be able to solve the SIS problem by outputting $(x - y) \mod q$, which is a short (i.e., polynomially bounded integer entry), non-zero vector in the kernel of A. Because the entries of y are bounded by q, it takes at most $\mathcal{O}(q^k \cdot \binom{m}{k}) \leq \mathrm{poly}(n)^k$ time for an adversary to try all k-sparse vectors $y \in \mathbb{Z}_\beta^m$. So it must be that for $k \in o(\frac{n}{\log n})$, such a k-sparse vector y does not exist with probability greater than $\mathrm{negl}(n)$, as otherwise an $o(2^n)$-time adversary would be able to find it by enumerating all candidates and use it to solve the SIS problem with non-negligible probability. □

We similarly have the following lemma:

Lemma 2.6. *Under Assumption 2.3, given a uniformly random matrix $A \in \mathbb{Z}_q^{n \times m}$ for $q, m, \beta \in \mathrm{poly}(n)$ and $q \geq n \cdot \beta$,*

Table 1. A summary of the bit complexities of our algorithms, as compared to the best known upper bounds for these problems in the white-box adversarial streaming model. Dash means that we provide the first algorithm for the problem in the white-box stream model. For k-sparse recovery, we require $k \geq n^c$ for an arbitrarily small constant $c > 0$.

PROBLEM	PREVIOUS SPACE	OUR SPACE	NOTE
K-SPARSE RECOVERY	–	$\tilde{\mathcal{O}}(k)$	DETECTS DENSE INPUT
L$_0$-NORM ESTIMATION	$\tilde{\mathcal{O}}(n^{1-\varepsilon+c\varepsilon})$	$\tilde{\mathcal{O}}(n^{1-\varepsilon})$	ACHIEVES n^{ϵ}-APPROXIMATION
LOW-RANK MATRIX RECOVERY	–	$\tilde{\mathcal{O}}(nk)$	DETECTS HIGH-RANK INPUT
LOW-RANK TENSOR RECOVERY	–	$\tilde{\mathcal{O}}(k(n_1 + \cdots + n_d))$	DETECTS HIGH CP-RANK INPUT
ROBUST PRINCIPLE COMPONENT ANAYSIS	–	$\tilde{\mathcal{O}}(nk + r)$	DETECTS NOT SPARSE + LOW RANK
RANK-DECISION	$\tilde{\mathcal{O}}(nk^2)$	$\tilde{\mathcal{O}}(nk)$	DETECTS HIGH-RANK INPUT
MAXIMUM MATCHING	–	$\tilde{\mathcal{O}}(nk)$	DETECTS LARGE MATCHING SIZE

if a matrix $X \in \mathbb{Z}_\beta^{\sqrt{m}\times\sqrt{m}}$ is generated by an $o(2^n)$-time adversary, then with probability at least $1 - negl(n)$, there does not exist a matrix $Y \in \mathbb{Z}_\beta^{\sqrt{m}\times\sqrt{m}}$ with $rank(Y) \leq k$, such that $X \neq Y \mod q$ and $Ax = Ay \mod q$, for x, y being the vectorizations of X and Y, respectively, and $k \in o(\frac{n}{\sqrt{m}\log n})$.

Proof. As in the proof of Lemma 2.4, an adversary is able to try all matrices $Y \in \mathbb{Z}_\beta^{\sqrt{m}\times\sqrt{m}}$ with $rank(Y) \leq k$ in $\mathrm{poly}(n)^{\sqrt{m}k}$ time. This is because there are $\mathcal{O}\binom{\sqrt{m}}{k}$ ways of positioning the linearly independent columns of Y, with $\mathrm{poly}(n)^{\sqrt{m}k}$ choices for their values when $\beta \in \mathrm{poly}(n)$. All remaining columns are linear combinations of the independent columns. Since there are $\mathrm{poly}(n)^k$-many possible combinations of coefficients and we choose $(\sqrt{m}$-$k)$ of them, there are $\mathrm{poly}(n)^{(\sqrt{m}-k)k}$ choices for the dependent columns. Therefore in total we have $\mathrm{poly}(n)^{\sqrt{m}k}$-many candidate matrices. For $k \in o(\frac{n}{\sqrt{m}\log n})$, there exists an $o(2^n)$-time adversary that is able to iterate through all candidate matrices. Thus, under Assumption 2.3, with overwhelming probability such a Y does not exist; otherwise, given X and Y, an adversary can easily solve the SIS problem by outputting $(x$-$y) \mod q$. □

3. Vector Recovery

3.1. k-Sparse Recovery Algorithm

Theorem 3.1. *Under Assumption 2.3, given a parameter $k \in \Theta(\frac{n^c}{\log n})$ for an arbitrary constant $c > 0$, and a length-n input vector with integer entries bounded by $\mathrm{poly}(n)$, there exists a streaming algorithm robust against $o(n^k)$ time-bounded white-box adversaries that determines if the input is k-sparse, and if so, recovers a k-sparse vector using $\tilde{\mathcal{O}}(k)$ bits of space in the random oracle model.*

Notation: A function $f(k)$ is said to be in $\omega(k)$ if for all real constants $c > 0$, there exists a constant $k_0 > 0$ such that $f(k) > c \cdot k$ for every $k \geq k_0$.

Algorithm 1 Recover-Vector(n, m, k)

```
Input: $m$ integer updates $u_t$ to a length-$n$ vector.
Let $f(k)$ be a function in $\omega(k)$ and $\tilde{\mathcal{O}}(k)$. Initialize a uniformly random matrix $A \in \mathbb{Z}_q^{(f(k)\cdot\log n)\times n}$ for $q \in \mathrm{poly}(n)$ and a zero vector $v$ of length $k \cdot \log n$.
for each update $u_t$ with $t \in [m]$ do
    Update $v$ by adding $u_t \cdot A_i$ to it, where $A_i$ is the $i^{th}$ column of $A$, and where the stream update changes the $i^{th}$ coordinate by an additive amount $u_t \in \mathbb{Z}_q$.
end for
for each $k$-sparse vector $y$ with entries $\in [-\beta, \beta]$ do
    if $Ay = v \mod q$ then
        return $y$
    end if
end for
return $None$
```

Proof. Algorithm 1 decides and recovers a k-sparse vector using $\tilde{\mathcal{O}}(k)$ bits. The algorithm receives a stream of integer updates to an underlying vector, whose entries are assumed to be at most $\beta \in poly(n)$ at any time. Thus we can interpret the stream updates to be mod q for $q, \beta \in \mathrm{poly}(n)$ and $q \geq n \cdot \beta$.

When an input vector x is k-sparse, for a uniformly random sketching matrix A, it is guaranteed by Lemma 2.4 that $Ay = Ax \mod q$ implies $y = x$. Therefore, in the k-sparse case, by enumerating over all k-sparse vectors, Algorithm 1 correctly recovers the input vector $y = x$. On the other hand, for inputs that have sparsity larger than k, Lemma 2.4 guarantees that during post-processing, the enumeration over k-sparse vectors will not find a vector y satisfying $Ay = v \mod q$. Thus, in this case Algorithm 1 outputs $None$ as desired. In the random oracle model, we can generate the columns of a uniformly random matrix A on the fly. Then, Algorithm 1 only stores a vector of length $f(k) \cdot \log n$ with entries bounded by $\mathrm{poly}(n)$, so $\tilde{\mathcal{O}}(k)$ bits of space. □

Remark 3.2. *With roughly k space, any white-box adversar-*

ially robust algorithm for k-sparse recovery has to assume that the adversary is at most n^k-time bounded. Otherwise, given that an algorithm using k words of memory has at most n^k states, for a k'-sparse input x with k' slightly larger than k, there exists an $x' \neq x$ that goes to the same state as x with high probability. Hence, the adversary would have enough time to find x and x'. If the adversary inserts either x or x' in the stream, followed by $-x$, the algorithm cannot tell if the input is 0 or $x' - x$. Thus, our algorithm is nearly optimal in the sense that it uses $\tilde{\mathcal{O}}(k)$ bits assuming the adversary is $o(2^{k \log n}) = o(n^k)$-time bounded.

3.2. Fast k-Sparse Recovery

Algorithm 1 enumerates over all possible k-sparse vectors in post-processing, which is time-inefficient. We now give a faster version of k-sparse recovery, which is also capable of identifying whether the input is k-sparse. In parallel we run an existing deterministic k-sparse recovery scheme that has fast update time assuming the input is k-sparse.

Theorem 3.3. *(Jafarpour, 2011) There exists a deterministic algorithm that recovers a k-sparse length-n vector in a stream using $\tilde{\mathcal{O}}(k)$ bits of space and $\mathrm{poly}(n)$ time.*

When the input vector is k-sparse, the algorithm in Theorem 3.3 outputs the input vector. However, when taking in an input vector with sparsity larger than k, this algorithm erroneously assumes the input to be k-sparse and has no guarantees, in which case the user cannot tell if the output is a correct recovery or not. To fix this, we run the two recovery schemes from Algorithm 1 and Theorem 3.3 in parallel. Any deterministic recovery scheme is robust against white-box adversaries, and therefore using the algorithm of Theorem 3.3 as a subroutine does not break our robustness.

Theorem 3.4. *Under Assumption 2.3, given a parameter $k \in \Theta(\frac{n^c}{\log n})$ for an arbitrary constant $c > 0$, and a length-n input vector with integer entries bounded by $\mathrm{poly}(n)$, there exists a streaming algorithm robust against $o(n^k)$ time-bounded white-box adversaries that determines if the input is k-sparse, and if so, recovers a k-sparse vector using $\tilde{\mathcal{O}}(k)$ bits of space and $\mathrm{poly}(n)$ time in the random oracle model.*

Proof. Algorithm 2 gives a fast version of k-sparse recovery. By running the two schemes in parallel, at the end of the stream, we can check the validity of its output as follows: if the fast algorithm recovers a vector y^* which is k-sparse and has the same SIS sketch as the input, i.e., ($Ay^* = v \mod q$), then by the correctness of Algorithm 1, y^* equals the input. On the other hand, if y^* is not k-sparse or its sketch does not match the SIS sketch v, then it must be that the input was not k-sparse. In both cases, Algorithm 2 is $\mathrm{poly}(n)$ time and returns the correct result. Both recovery schemes from Algorithm 1 and Theorem 3.3 use $\tilde{\mathcal{O}}(k)$ bits of space. Thus, Algorithm 2 uses $\tilde{\mathcal{O}}(k)$ bits as well. Evaluating the output of the fast recovery scheme ($eval(\mathcal{F}(\cdot))$) and comparing the sketches takes $\mathrm{poly}(n)$ time, so the entire algorithm takes $\mathrm{poly}(n)$ time. □

Algorithm 2 Fast-Recover(n, m, k)

```
  Input: $m$ integer updates $u_t$ to a length-$n$ vector.
  Initiate an instance of the fast $k$-sparse recovery scheme $\mathcal{F}(\cdot)$ from Theorem 3.3.
  Let $f(k)$ be a function in $\omega(k)$ and $\tilde{\mathcal{O}}(k)$. Initialize a uniformly random matrix $A \in \mathbb{Z}_q^{(f(k)\cdot\log n)\times n}$ for $q \in \mathrm{poly}(n)$ and a zero vector $v$ of length $k \cdot \log n$.
  for each update $u_t$ with $t \in [m]$ do
    Feed the update to the initiated instance $\mathcal{F}(\cdot)$.
    Update $v$ by adding $u_t \cdot A_i$ to it, where $A_i$ is the $i^{th}$ column of $A$, and where the stream update changes the $i^{th}$ coordinate by an additive amount $u_t \in \mathbb{Z}_q$.
  end for
  $y^* \leftarrow eval(\mathcal{F}(\cdot))$
  if $y^*$ is $k$-sparse andalso $\|y^*\|_\infty \leq \beta$ andalso $Ay^* = v \mod q$ then
    return $y^*$
  else
    return $None$
  end if
```

3.3. Applications of k-Sparse Recovery

3.3.1. Estimating the ℓ_0 Norm

Using our k-sparse recovery algorithm as a subroutine, we can construct an efficient ℓ_0 estimation algorithm. This algorithm gives an n^ε-approximation to the ℓ_0 norm of a vector, whose entries are assumed to be bounded by $\mathrm{poly}(n)$.

Theorem 3.5. *Under Assumption 2.3, for constant $\varepsilon < 1$, there exists a streaming algorithm robust against $o(n^{n^{1-\varepsilon}})$ time-bounded white-box adversaries, which estimates the L_0 norm of a length-n vector in the stream within a multiplicative factor of n^ϵ using $\tilde{\mathcal{O}}(n^{1-\epsilon})$ bits of space and $\mathrm{poly}(n)$ time in the random oracle model.*

Algorithm 3 Estimate-L0(n, m, ε)

```
  Input: $m$ integer updates $u_t$ to a length-n vector.
  $result \leftarrow$ Fast-recover(n, m, n$^{1-\varepsilon}$)
  if $result = None$ then
    return n$^{1-\varepsilon}$
  else
    return $\ell_0(result)$
  end if
```

Proof. Algorithm 3 gives an n^ε-approximation to the ℓ_0 norm using Algorithm 1 as a subroutine. Given a parameter $\varepsilon > 0$, we set the parameter k for k-sparse recovery to be $n^{1-\varepsilon}$. The space used by Algorithm 3 is then $\tilde{\mathcal{O}}(n^{1-\varepsilon})$ bits. Also, both the recovery and the post-processing run in $\mathrm{poly}(n)$ time, so estimation can be done in $\mathrm{poly}(n)$ time. For correctness, if the vector is $n^{1-\varepsilon}$-sparse, it can be recovered perfectly, and thus $\ell_0(result)$ is the exact value of the ℓ_0 norm. Otherwise, for an input vector with more than $n^{1-\varepsilon}$ non-zero entries, its ℓ_0 norm lies in the range $(n^{1-\varepsilon}, n]$. Hence, if we estimate its norm to be $n^{1-\varepsilon}$, this gives an n^{ε}-approximation. □

4. Matrix Recovery

4.1. Low-Rank Matrix Recovery

In addition to recovering sparse vectors, we can recover low-rank matrices. We propose a white-box adversarially robust algorithm for the low-rank matrix recovery problem, which is efficient in terms of both time and space.

Similar to the k-sparse vector recovery problem, in order to achieve a fast update time while ensuring that the algorithm correctly detects inputs with rank larger than expected, we run two matrix recovery schemes in parallel. We will maintain one sketch based on a uniformly random matrix to distinguish if the input rank is too high to recover, and the other sketch will allow us to recover the input matrix if it is promised to be low rank.

Theorem 4.1. *(Recht et al., 2010) Let $\alpha = O(nk \log n)$ and let A be a random matrix of dimension $\alpha \times n^2$, with entries sampled from an i.i.d. symmetric Bernoulli distribution:*

$$A_{ij} = \begin{cases} \sqrt{\frac{1}{\alpha}} & \textit{with probability } \frac{1}{2} \\ -\sqrt{\frac{1}{\alpha}} & \textit{with probability } \frac{1}{2} \end{cases}$$

Interpret A as a linear map $\mathcal{A} : \mathbb{R}^{n \times n} \to \mathbb{R}^{\alpha}$ that computes Ax for x being the vectorization of an input $X \in \mathbb{R}^{n \times n}$. Then, given a rank-$r$ matrix $X_0 \in \mathbb{R}^{n \times n}$ and $b = \mathcal{A}(X_0)$ for $1 \leq r \leq min(k, n/2)$, with high probability X_0 is the unique low-rank solution to $\mathcal{A}(X) = b$ satisfying $rank(X) \leq r$. Moreover, X_0 can be recovered by solving a convex program: $argmin_X \|X\|_*$ subject to $\mathcal{A}(X) = b$.

We state our main theorem for matrix recovery:

Theorem 4.2. *Under Assumption 2.3, given an integer parameter k, there exists a streaming algorithm robust against $o(n^{nk})$ time-bounded white-box adversaries that either states that the input matrix has rank greater than k, or recovers the input matrix with rank at most k using $\tilde{\mathcal{O}}(nk)$ bits of space and $\mathrm{poly}(n)$ time in the random oracle model.*

Proof. Algorithm 4 decides and recovers a matrix with rank no greater than k using $\tilde{\mathcal{O}}(nk)$ bits of space. For any input matrix $X \in \mathbb{Z}_{\beta}^{n \times n}$ with $\beta \in \mathrm{poly}(n)$, $q \geq n \cdot \beta$, and $rank(X) \leq k$, by the uniqueness of the low-rank solution given in Theorem 4.1, X can be recovered by solving a convex program, and its product with the matrix H matches the sketch v. On the other hand, when $rank(X) > k$, by Lemma 2.6 under the SIS hardness assumption, there does not exist a low-rank matrix Y distinct from X, for which $Hy = v = Hx \mod q$ with x, y being the vectorization of X, Y, respectively. Therefore, in this case Algorithm 4 outputs $None$, as desired.

Algorithm 4 Recover-Matrix(n, m, k)

```
Input: $m$ integer updates $u_t$ to an $n \times n$ matrix.
Let $f(k)$ be a function in $\omega(k)$ and $\tilde{\mathcal{O}}(k)$. Initialize a uniformly random matrix $H \in \mathbb{Z}_q^{f(k) \cdot n \log n \times n^2}$ for $q \in \mathrm{poly}(n)$, a matrix $A : \alpha \times n^2$ as specified in Theorem 4.1, and zero vectors $v, w$ of length $f(k) \cdot n \log n$.
for each update $u_t$ with $t \in [m]$ do
    Update $v$ by adding $u_t \cdot H_i$ to it, and update $w$ by adding $u_t \cdot A_i$ to it, where $i$ corresponds to the vectorized index of the update, and where $H_i, A_i$ are the $i^{th}$ columns of $H, A$, respectively.
end for
$X_0 \leftarrow argmin_X \|X\|_*$ subject to $A \cdot vectorize(X) = w$
if $rank(X_0) \leq k$ andalso $X_0 \in \mathbb{Z}_{\beta}^{n \times n}$ andalso $H \cdot vectorize(X_0) = v \mod q$ then
    return $X_0$
else
    return $None$
end if
```

Both random matrices H and A used in Algorithm 4 can be generated on the fly in the random oracle model. Therefore, the recovery algorithm only stores two sketch vectors of length $\tilde{\mathcal{O}}(nk)$ with entries bounded by $\mathrm{poly}(n)$, taking $\tilde{\mathcal{O}}(nk)$ bits in total. Solving the convex problem with the ellipsoid method and then comparing the solution with the sketch is $\mathrm{poly}(n)$ time, giving overall $\mathrm{poly}(n)$ time. □

Remark 4.3. *We argue the optimality of our low-rank matrix recovery algorithm: with roughly nk space, any white-box adversarially robust algorithm for low-rank matrix recovery has to assume that the adversary is n^{nk}-time bounded. Otherwise, the adversary has enough time to find a pair of inputs $X \neq X'$ that go to the same state and satisfy $rank(X' - X) > k$. Inserting X then $-X$, or X' then $-X$ into the stream, the algorithm cannot tell if the input is 0 or $X' - X$. Hence, our $\tilde{\mathcal{O}}(nk)$-bit algorithm assuming an $o(2^{nk \log n}) = o(n^{nk})$ adversary is nearly optimal.*

4.2. Applications of Low-Rank Matrix Recovery

Our low-rank matrix recovery algorithm can be applied to a number of other problems on data streams.

4.2.1. Rank Decision Problem

Definition 4.4. *(Rank Decision Problem). Given an integer k, and an $n \times n$ matrix A, determine whether the rank of A is larger than k.*

Theorem 4.5. *Under Assumption 2.3, given an integer parameter k, there exists a streaming algorithm robust against $o(n^{nk})$ time-bounded white-box adversaries that solves the rank decision problem using $\tilde{O}(nk)$ bits of space and $\mathrm{poly}(n)$ time in the random oracle model.*

Proof. This problem is solved by running Algorithm 4 with parameter k. This directly improves (Ajtai et al., 2022). □

4.2.2. Graph Matching

Definition 4.6. *(Maximum Matching Problem) Given an undirected graph $G = (V, E)$, the maximum matching problem is to find a maximum set of vertex disjoint edges in G. In a stream, we see insertions and deletions to edges.*

Theorem 4.7. *Under Assumption 2.3, given an integer upper bound k' on the size of a maximum matching in the graph, there is a streaming algorithm robust against $o(n^{2nk'})$ time-bounded white-box adversaries that finds a maximum matching in a graph using $\tilde{O}(nk')$ bits of space and $\mathrm{poly}(n)$ time in the random oracle model.*

Proof. We can use the fact that the rank of the $n \times n$ Tutte matrix A of the graph G, where $A_{i,j} = 0$ if there is no edge from i to j, and $A_{i,j} = x_{i,j}$ and $A_{j,i} = -x_{i,j}$ for an indeterminate $x_{i,j}$ otherwise, equals twice the maximum matching size of G. Here the rank of A is defined to be the maximum rank of A over the reals over all assignments to its indeterminates. The main issue, unlike standard algorithms (see, e.g., Sections 4.2.1 and 4.2.2 of (Cheung et al., 2013)), is that we cannot fill in the entries of A randomly in a stream in the white-box model because the adversary can see our state and try to fool us. Fortunately, there is a fix - in the stream we replace all $x_{i,j}$ deterministically with the number 1. Call this deterministically filled in matrix A', and note that the rank of A' is at most the rank of A, and the latter is twice the maximum matching size. We then run our low-rank matrix recovery algorithm with parameter k set to $2k'$. If we detect that the rank of A' is greater than $2k'$, then the rank of A is greater than $2k'$, and the maximum matching size is larger than k' and we stop the algorithm and declare this. Otherwise, we have successfully recovered A' and now that the stream is over, the locations of the 1s are exactly the indeterminates in A, and so we have recovered A and hence G and thus can run any offline algorithm for computing a maximum matching of G. □

4.3. Extension to Robust PCA and Tensors

The problem of Robust Principal Component Analysis is defined as follows:

Definition 4.8. *(Robust Principal Component Analysis) Consider a data matrix $M \in \mathbb{Z}_q^{n\times n}$ for $q \geq \mathrm{poly}(n)$, such that there exists a decomposition $M = L + S$, where $L \in \mathbb{Z}_q^{n\times n}$ satisfies $rank(L) \leq k$ and $S \in \mathbb{Z}_q^{n\times n}$ has at most r non-zero entries. The robust principal component analysis (RPCA) problem seeks to find the components L and S.*

A recurring idea in our algorithms is to run two algorithms in parallel: (1) an algorithm to detect if the input is drawn from a small family of inputs, such as those which are sparse or low rank or both, and (2) a time-efficient deterministic algorithm which recovers the input if it is indeed drawn from such a family. The algorithm in (1) relies on the hardness of SIS while the algorithm in (2) is any time and space efficient deterministic, and thus white box adversarially robust, algorithm. For (1) we use an SIS matrix, and for (2), for robust PCA we use the algorithm of (Tanner & Vary, 2020) while for tensors we use the algorithm of (Grotheer et al., 2019). See Appendix A and Appendix B for details.

5. Conclusion

We give robust streaming algorithms against computationally bounded white-box adversaries under cryptographic assumptions. We design efficient recovery algorithms for vectors, matrices, and tensors which can detect if the input is not sparse or low rank. We use these to improve upon and solve new problems in linear algebra and optimization, such as detecting and finding a maximum matching if it is small. It would be interesting to explore schemes that can recover vectors that are only approximately k-sparse or matrices that are only approximately rank-k. We make progress on the latter by considering robust PCA, but there is much more to be done. Also, although our algorithm improves the space-accuracy trade-off for ℓ_0-norm estimation, it is unclear if it is optimal, and it would be good to generalize to ℓ_p norms for $p > 0$, as well as other statistics of a vector.

Acknowledgment: We thank Aayush Jain for a helpful discussion on pseudorandom functions and the random oracle model. D. Woodruff would like to thank support from the National Institute of Health (NIH) grant 5R01 HG 10798-2 and a Simons Investigator Award.

References

Aggarwal, D., Dadush, D., Regev, O., and Stephens-Davidowitz, N. Solving the shortest vector problem in 2n time using discrete gaussian sampling: Extended abstract. In *Proceedings of the Forty-Seventh Annual*

ACM Symposium on Theory of Computing, STOC '15, pp. 733–742, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450335362. doi: 10.1145/2746539.2746606. URL https://doi.org/10.1145/2746539.2746606.

Ajtai, M., Braverman, V., Jayram, T., Silwal, S., Sun, A., Woodruff, D. P., and Zhou, S. The white-box adversarial data stream model. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '22, pp. 15–27, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392600. doi: 10.1145/3517804.3526228. URL https://doi.org/10.1145/3517804.3526228.

Alon, N., Matias, Y., and Szegedy, M. The space complexity of approximating the frequency moments. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pp. 20–29, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917855. doi: 10.1145/237814.237823. URL https://doi.org/10.1145/237814.237823.

Alon, N., Matias, Y., and Szegedy, M. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

Alon, N., Ben-Eliezer, O., Dagan, Y., Moran, S., Naor, M., and Yogev, E. Adversarial laws of large numbers and optimal regret in online classification. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 447–455, 2021.

Attias, I., Cohen, E., Shechner, M., and Stemmer, U. A framework for adversarial streaming via differential privacy and difference estimators. *CoRR*, abs/2107.14527, 2021.

Banerjee, A., Peikert, C., and Rosen, A. Pseudorandom functions and lattices. volume 2011, pp. 401, 01 2011. ISBN 978-3-642-29010-7. doi: 10.1007/978-3-642-29011-4_42.

Bellare, M. and Rogaway, P. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pp. 62–73, New York, NY, USA, 1993. Association for Computing Machinery. ISBN 0897916298. doi: 10.1145/168588.168596. URL https://doi.org/10.1145/168588.168596.

Bellare, M. and Rogaway, P. The exact security of digital signatures-how to sign with rsa and rabin. In Maurer, U. (ed.), *Advances in Cryptology — EUROCRYPT '96*, pp. 399–416, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. ISBN 978-3-540-68339-1.

Ben-Eliezer, O. and Yogev, E. The adversarial robustness of sampling. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS*, pp. 49–62, 2020.

Ben-Eliezer, O., Jayaram, R., Woodruff, D. P., and Yogev, E. A framework for adversarially robust streaming algorithms. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS'20, pp. 63–80, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371087. doi: 10.1145/3375395.3387658. URL https://doi.org/10.1145/3375395.3387658.

Ben-Eliezer, O., Jayaram, R., Woodruff, D. P., and Yogev, E. A framework for adversarially robust streaming algorithms. *SIGMOD Rec.*, 50(1):6–13, 2021.

Ben-Eliezer, O., Eden, T., and Onak, K. Adversarially robust streaming via dense-sparse trade-offs. In *5th Symposium on Simplicity in Algorithms, SOSA*, 2022.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Srndic, N., Laskov, P., Giacinto, G., and Roli, F. Evasion attacks against machine learning at test time. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD, Proceedings, Part III*, pp. 387–402, 2013.

Braverman, V., Hassidim, A., Matias, Y., Schain, M., Silwal, S., and Zhou, S. Adversarial robustness of streaming algorithms through importance sampling. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2021.

Brown, P., Haas, P. J., Myllymaki, J., Pirahesh, H., Reinwald, B., and Sismanis, Y. 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*, pp. 161–180, 2005.

Candès, E. J. and Wakin, M. B. An introduction to compressive sampling. *IEEE Signal Process. Mag.*, 25(2):21–30, 2008. doi: 10.1109/MSP.2007.914731. URL https://doi.org/10.1109/MSP.2007.914731.

Candès, E. J., Li, X., Ma, Y., and Wright, J. Robust principal component analysis? *J. ACM*, 58 (3), jun 2011. ISSN 0004-5411. doi: 10.1145/1970392.1970395. URL https://doi.org/10.1145/1970392.1970395.

Canetti, R., Goldreich, O., and Halevi, S. The random oracle methodology, revisited. *J. ACM*, 51(4):

557–594, jul 2004. ISSN 0004-5411. doi: 10.1145/1008731.1008734. URL https://doi.org/10.1145/1008731.1008734.

Chakrabarti, A., Ghosh, P., and Stoeckl, M. Adversarially robust coloring for graph streams. In *13th Innovations in Theoretical Computer Science Conference, ITCS*, 2022.

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

Chan, T. M. and He, Q. More dynamic data structures for geometric set cover with sublinear update time. In *37th International Symposium on Computational Geometry, SoCG*, pp. 25:1–25:14, 2021.

Chandrasekaran, V., Sanghavi, S., Parrilo, P. A., and Willsky, A. S. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 21(2):572–596, 2011. doi: 10.1137/090761793. URL https://doi.org/10.1137/090761793.

Cheung, H. Y., Kwok, T. C., and Lau, L. C. Fast matrix rank algorithms and applications. *J. ACM*, 60(5):31:1–31:25, 2013.

Clifford, P. and Cosma, I. A. A simple sketching algorithm for entropy estimation over streaming data. In *International Conference on Artificial Intelligence and Statistics*, 2013.

Cubuk, E. D., Zoph, B., Mané, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018.

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

Driscoll, J. R., Sarnak, N., Sleator, D. D., and Tarjan, R. E. Making data structures persistent. *J. Comput. Syst. Sci.*, 38(1):86–124, 1989.

Fiat, A. and Kaplan, H. Making data structures confluently persistent. *J. Algorithms*, 48(1):16–58, 2003.

Ganguly, S. and Majumder, A. Deterministic k-set structure. In Vansummeren, S. (ed.), *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*, pp. 280–289. ACM, 2006.

Goldreich, O., Goldwasser, S., and Micali, S. How to construct random functions. *J. ACM*, 33(4):792–807, aug 1986. ISSN 0004-5411. doi: 10.1145/6490.6503. URL https://doi.org/10.1145/6490.6503.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014. URL http://arxiv.org/abs/1412.6572.

Grotheer, R., Li, S., Ma, A., Needell, D., and Qin, J. Iterative hard thresholding for low cp-rank tensor models. 08 2019.

Hassidim, A., Kaplan, H., Mansour, Y., Matias, Y., and Stemmer, U. Adversarially robust streaming algorithms via differential privacy. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020.

Huang, S. H., Papernot, N., Goodfellow, I. J., Duan, Y., and Abbeel, P. Adversarial attacks on neural network policies. In *5th International Conference on Learning Representations, ICLR*, 2017.

Ilyas, A., Engstrom, L., and Madry, A. Prior convictions: Black-box adversarial attacks with bandits and priors. *CoRR*, abs/1807.07978, 2018.

Jafarpour, S. *Deterministic Compressed Sensing*. PhD thesis, Princeton University, 2011.

Jayaram, R. and Woodruff, D. P. Towards optimal moment estimation in streaming and distributed models. *ACM Trans. Algorithms*, may 2023. ISSN 1549-6325. doi: 10.1145/3596494. URL https://doi.org/10.1145/3596494. Just Accepted.

Kaplan, H. Persistent data structures. In *Handbook of Data Structures and Applications*. Chapman and Hall/CRC, 2004.

Kaplan, H., Mansour, Y., Nissim, K., and Stemmer, U. 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*, pp. 94–121, 2021.

Karney, C. F. F. Sampling exactly from the normal distribution. *ACM Trans. Math. Softw.*, 42(1), jan 2016. ISSN 0098-3500. doi: 10.1145/2710016. URL https://doi.org/10.1145/2710016.

Kim, S. B. *Pseudorandom Functions with New Properties from Hard Lattice Problems*. PhD thesis, 2021. - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; - 2023-03-05.

Koblitz, N. and Menezes, A. The random oracle model: A twenty-year retrospective. Cryptology ePrint Archive, Paper 2015/140, 2015. URL https://eprint.iacr.org/2015/140. https://eprint.iacr.org/2015/140.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial machine learning at scale. In *5th International Conference on Learning Representations, ICLR, Conference Track Proceedings*, 2017.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Liu, Y., Chen, X., Liu, C., and Song, D. Delving into transferable adversarial examples and black-box attacks. In *5th International Conference on Learning Representations, ICLR, Conference Track Proceedings*, 2017.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR, Conference Track Proceedings*, 2018.

Menuhin, B. and Naor, M. Keep that card in mind: Card guessing with limited memory. *CoRR*, abs/2107.03885, 2021.

Micciancio, D. and Peikert, C. Hardness of sis and lwe with small parameters. Cryptology ePrint Archive, Paper 2013/069, 2013. URL `https://eprint.iacr.org/2013/069`. `https://eprint.iacr.org/2013/069`.

Padmanabhan, S., Bhattacharjee, B., Malkemus, T., Cranston, L., and Huras, M. Multi-dimensional clustering: A new data layout scheme in DB2. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pp. 637–641, 2003.

Recht, B., Fazel, M., and Parrilo, P. A. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010. doi: 10.1137/070697835. URL `https://doi.org/10.1137/070697835`.

Roghani, M., Saberi, A., and Wajc, D. Beating the folklore algorithm for dynamic matching. In *13th Innovations in Theoretical Computer Science Conference, ITCS*, 2022.

Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., and Madry, A. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS.*, pp. 5019–5031, 2018.

Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., and Price, T. G. Access path selection in a relational database management system. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, pp. 23–34. ACM, 1979.

Shukla, A., Deshpande, P., Naughton, J. F., and Ramasamy, K. Storage estimation for multidimensional aggregates in the presence of hierarchies. In *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases*, pp. 522–531, 1996.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. *International Conference on Learning Representations*, 2014.

Tanner, J. and Vary, S. Compressed sensing of low-rank plus sparse matrices. *ArXiv*, abs/2007.09457, 2020.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I. J., Boneh, D., and McDaniel, P. D. Ensemble adversarial training: Attacks and defenses. In *6th International Conference on Learning Representations, ICLR, Conference Track Proceedings*, 2018.

Wajc, D. Rounding dynamic matchings against an adaptive adversary. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pp. 194–207, 2020.

Woodruff, D. P. and Zhou, S. Tight bounds for adversarially robust streams and sliding windows via difference estimators. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pp. 1183–1196, 2021.

A. Robust Principal Component Analysis

As in Section 2.2, we derive the following lemma based on the hardness of the SIS problem.

Lemma A.1. *Under Assumption 2.3, given a uniformly random matrix $A \in \mathbb{Z}_q^{n\times m}$ for $q, m, \beta \in \mathrm{poly}(n)$ and $q \geq n \cdot \beta$, if a matrix $X \in \mathbb{Z}_\beta^{\sqrt{m}\times\sqrt{m}}$ is generated by an $o(2^n)$-time adversary, then with probability $\geq 1 - negl(n)$, there do not exist matrices $L, S \in \mathbb{Z}_\beta^{\sqrt{m}\times\sqrt{m}}$ with $rank(L) \leq k$ and*[4] *$nnz(S) \leq r$, for which $X \neq L+S \mod q$ and $Ax = A(l+s) \mod q$, for x, l, s being the vectorization of X, L and S, respectively, and r, k satisfying $k \in o(\frac{n-r\log n}{\sqrt{m}\log n})$.*

Proof. Similar to the proof of Lemma 2.6, an adversary is able to try all pairs of matrices $L, S \in \mathbb{Z}_\beta^{\sqrt{m}\times\sqrt{m}}$ with $rank(L) \leq k$ and $nnz(S) \leq r$ in $\mathrm{poly}(n)^{r+k\sqrt{m}}$ time. As shown in the proof of Lemma 2.6, there are $\mathrm{poly}(n)^{\sqrt{m}k}$-many candidates for L. For the sparse matrix S, there are $\binom{m}{r} \in \mathrm{poly}(n)^r$ ways of positioning the non-zero entries, with their values chosen in $\mathrm{poly}(n)$. Therefore in total there are $\mathrm{poly}(n)^{r+k\sqrt{m}}$ pairs of candidate matrices.

[4] We use $nnz(S)$ to denote the number of non-zero entries of a matrix S.

When $k \in o(\frac{n-r\log n}{\sqrt{m}\log n})$, there exists an $o(2^n)$-time adversary that is able to iterate through all candidate pairs. Thus, under Assumption 2.3, with overwhelming probability such an L, S do not exist, otherwise, given L and S, an adversary can solve the SIS problem by outputting $(x\text{-}l\text{-}s) \mod q$. □

As in our matrix recovery algorithm, we run a compressed sensing scheme for RPCA in parallel to achieve a fast recovery time. This fast scheme approximates a unique pair of low rank and sparse matrices from their sum, assuming the sum is decomposable into a pair of such matrices.

Theorem A.2. *(Tanner & Vary, 2020) Let $\alpha = O((nk+r)\cdot\log n)$, and let A be a random matrix of dimension $\alpha \times n^2$, with entries sampled from an i.i.d. symmetric Bernoulli distribution:*

$$A_{ij} = \begin{cases} \sqrt{\frac{1}{\alpha}} & \textit{with probability } \frac{1}{2} \\ -\sqrt{\frac{1}{\alpha}} & \textit{with probability } \frac{1}{2} \end{cases}$$

Interpret A as a linear map $\mathcal{A} : \mathbb{R}^{n\times n} \to \mathbb{R}^{\alpha}$ which computes Ax for x being the vectorization of an input $X \in \mathbb{R}^{n\times n}$. Then given $b = \mathcal{A}(L_0 + S_0)$, with high probability, L_0, S_0 is the unique solution to $\mathcal{A}(L + S) = b$ satisfying $rank(L_0) \le k$ and $nnz(S) \le r$. Moreover, L_0, S_0 can be recovered efficiently to a precision of $\|(L + S) - (L_0 + S_0)\|_F \le 42\varepsilon$ by solving a semidefinite program:

$$argmin_{L,S}(\|L\|_* + \sqrt{2r/s}\cdot\|S\|_1)$$

subject to

$$\|\mathcal{A}(L+S) - b\|_2 \le \varepsilon$$

with the nuclear norm $\|\cdot\|_*$ of a matrix M defined as the sum of its singular values $\|M\|_* = \sum_i \sigma_i(M)$; and the 1-norm $\|\cdot\|_1$ defined as its maximum absolute column sum $\|M\|_1 = max_{0\le j\le n}\sum_{i=1}^{n}|M_{ij}|$.

Note that for an integer stream, we can set the error parameter $\varepsilon \le \frac{1}{\mathrm{poly}(n)}$ and then round the entries of the result to integers to guarantee exact recovery.

Proof. (of Theorem 1.5) Algorithm 5 determines if an $n\times n$ input matrix can be decomposed into a low rank matrix plus a sparse matrix. For any input matrix $X_0 = L_0 + S_0 \in \mathbb{Z}_\beta^{n\times n}$ with $rank(L_0) \le k$ and $nnz(S_0) \le r$, by the uniqueness of the solution pair given in Theorem A.2, L_0, S_0 can be recovered by solving a semidefinite program, and the product of their sum with the matrix H matches the sketch v. On the other hand, when the input X cannot be decomposed into low rank and sparse components, by Lemma A.1 under the SIS hardness assumption, there does not exist a pair of low rank and sparse matrices L', S' such that $X \neq L'+S'$ and $H(l'+s') = v = Hx \mod q$, for l', s', x being the vectorization of L', S', X, respectively. Therefore, in this case Algorithm 5 outputs $None$, as desired.

Algorithm 5 **RPCA**(n, m, k, r)

```
Input: $m$ integer updates $u_t$ to an $n \times n$ matrix.
Let $f(k)$ be a function in $\omega(k)$ and $\tilde{O}(k)$. Initialize a uniformly random matrix $H \in \mathbb{Z}_q^{(f(k)\cdot n+r)\log n\times n^2}$ for $q \in \mathrm{poly}(n)$, a fast recovery matrix $A : (nk+r)\log n \times n^2$ as specified in A.2, and zero vectors $v, w$ of length $(f(k)\cdot n + r)\log n$.
for each update $u_t$ with $t \in [m]$ do
    Update $v$ by adding $u_t \cdot H_i$ to it, and update $w$ by adding $u_t \cdot A_i$ to it, where $i$ corresponds to the vectorized index of the update, and where $H_i, A_i$ are the $i^{th}$ columns of $H, A$, respectively.
end for
$L_0, S_0 \leftarrow argmin_{L,S}(\|L\|_* + \sqrt{2r/s}\cdot\|S\|_1)$ subject to $\|\mathcal{A}(L+S) - b\|_2 \le \frac{1}{\mathrm{poly}(n)}$.
if $rank(L_0) \le k$ andalso $nnz(S_0) \le r$ andalso $L_0, S_0 \in \mathbb{Z}_\beta^{n\times n}$ andalso $H \cdot vectorize(L_0 + S_0) = v \mod q$ then
    return $L_0, S_0$
else
    return $None$
end if
```

Both random matrices H and A used in Algorithm 5 can be generated on the fly in the random oracle model. Therefore, the recovery algorithm only stores two sketch vectors of length $\tilde{\mathcal{O}}(nk + r)$ with entries bounded by $\mathrm{poly}(n)$, taking $\tilde{\mathcal{O}}(nk + r)$ bits in total. Also, solving the semidefinite program and then comparing the solution with the sketch takes $\mathrm{poly}(n)$ time, giving overall $\mathrm{poly}(n)$ time. □

B. Tensor Recovery

Similar to our vector and matrix recovery algorithms, we propose an algorithm which recovers tensors with low CANDECOMP/PARAFAC (CP) rank.

Notation: Let $\otimes$ denote the outer product of two vectors. Then one can build a rank-1 tensor in $\mathbb{Z}^{n_1\times n_2\times\cdots\times n_d}$ by taking the outer product $x_1 \otimes x_2 \otimes \cdots \otimes x_d$ where $x_i \in \mathbb{Z}^{n_i}$.

Definition B.1. *(CP-rank) For a tensor $X \in \mathbb{Z}_q^{n_1\times\cdots\times n_d}$, consider it to be the sum of r rank-1 tensors: $X = \sum_{i=1}^{r}(x_{i1} \otimes x_{i2} \otimes \cdots \otimes x_{id})$ where $x_{ij} \in \mathbb{Z}_q^{n_j}$. The smallest number of rank-1 tensors that can be used to express a tensor X is then defined to be the rank of the tensor.*

As in Section 2.2, we derive the following lemma based on the hardness of the SIS problem.

Lemma B.2. *Under Assumption 2.3, given a uniformly random matrix $A \in \mathbb{Z}_q^{n\times m}$ for $q, m, \beta \in \mathrm{poly}(n)$ and $q \geq$*

$n \cdot \beta$, if a tensor $X \in \mathbb{Z}_\beta^{n_1\times\cdots\times n_d}$ is generated by an $o(2^n)$-time adversary, where $\prod n_i = m$, then with probability $\geq 1-negl(n)$, there does not exist a tensor $Y \in \mathbb{Z}_\beta^{n_1\times\cdots\times n_d}$ with $rank(Y) \leq k$, such that $X \neq Y \mod q$ and $Ax = Ay \mod q$, for x, y being the vectorization of X and Y, respectively, $\prod n_i = m$, and $k \in o(\frac{n}{(n_1+\cdots+n_d)\log n})$.

Proof. Similar to the proof of Lemma 2.6, an adversary is able to try all low rank tensors $Y \in \mathbb{Z}_\beta^{n_1\times\cdots\times n_d}$ with $rank(Y) \leq k$ in $\mathrm{poly}(n)^{k(n_1+\cdots+n_d)}$ time. For each x_{ij}, there are $\mathrm{poly}(n)^{n_j}$ choices of its value. So we have $\mathrm{poly}(n)^{n_1+\cdots+n_d}$ many possible rank-1 tensors. Choosing k-many of them to generate a rank-k tensor, that is $\mathrm{poly}(n)^{k(n_1+\cdots+n_d)}$ candidates in total.

When $k \in o(\frac{n}{(n_1+\cdots+n_d)\log n})$, there exists an $o(2^n)$-time adversary that is able to iterate through all candidate pairs. Thus, under Assumption 2.3, with overwhelming probability such a Y does not exist; otherwise, given L and S, an adversary can solve the SIS problem by outputting $(x\text{-}y) \mod q$. □

As we did for vector and matrix recovery problems, we can run a fast low rank tensor estimation scheme in parallel in our tensor recovery algorithm.

Theorem B.3. *(Grotheer et al., 2019) Let $\mathcal{A} : \mathbb{R}^{n_1\times\cdots\times n_d} \to \mathbb{R}^{k(n_1+\cdots n_d)logn}$ be a Gaussian measurement operator whose entries are properly normalized, i.i.d. Gaussian random variables. Then given a measurement $b = \mathcal{A}(X)$ for a tensor $X \in \mathbb{R}^{n1\times\cdots\times n_d}$, there exists an algorithm that gives an estimate $X_0 \in \mathbb{R}^{n_1\times\cdots\times n_d}$ with $\|X_0 - X\|_F \leq \frac{1}{\mathrm{poly}(n)}$ with high probability using $\mathrm{poly}(n)$ time, for $n = \prod_1^d n_i$.*

Remark B.4. *For our purposes, we round the Gaussian random variables to additive integer multiples of $\frac{1}{poly(n)}$. This rounding changes the norm of the measurement by at most additive $\frac{1}{poly(n)}$, and therefore asymptotically does not change the result in Theorem B.3. The discretized random variables can then be constructed to the desired precision using uniformly random bits (Karney, 2016) generated by a random oracle.*

Then running the algorithm from Theorem B.3 in a stream, we only have to maintain and update a measurement vector of length $\tilde{O}(k(n_1+\cdots n_d))$ with entries bounded in $\mathrm{poly}(n)$, giving an overall $\tilde{O}(k(n_1+\cdots+n_d))$ bits of space usage.

For an integer stream, we can round the entries of the estimation result to integers to guarantee exact recovery. We formulate the proof for the low rank tensor recovery algorithm as follows.

Algorithm 6 Recover-tensor($n_1, \cdots n_d$, m, k)

```
Input: $m$ integer updates $u_t$ to an $n_1 \times \cdots \times n_d$ tensor.
Initiate an instance of the fast low rank tensor recovery scheme $\mathcal{F}(\cdot)$ from Theorem A.2.
Let $n = \prod_1^d n_i$. Let $f(k)$ be a function in $\omega(k)$ and $\tilde{O}(k)$. Initialize a uniformly random matrix $H \in \mathbb{Z}_q^{f(k)(n_1+\cdots+n_d)\log n \times n}$ for $q \in \mathrm{poly}(n)$, and a zero vector $v$ of length $f(k)(n_1+\cdots+n_d)\log n$
for each update $u_t$ with $t \in [m]$ do
    Feed the update to the initiated instance $\mathcal{F}(\cdot)$.
    Update $v$ by adding $u_t \cdot A_i$ to it, where $H_i$ is the $i^{th}$ column of $H$, and where the stream update changes the $i^{th}$ coordinate by an additive amount $u_t \in \mathbb{Z}_q$.
end for
$X^* \leftarrow eval(\mathcal{F}(\cdot))$
if $rank(X^*) \leq k$ andalso $X^* \in \mathbb{Z}_\beta^{n_1\times\cdots\times n_d}$ andalso $H \cdot vectorize(X^*) = v \mod q$ then
    return $X^*$
else
    return $None$
end if
```

Proof. (of Theorem 1.6) Algorithm 6 determines if an $n_1 \times n_2 \times \cdots \times n_d$ input tensor has rank at most k and if so, recovers the input tensor.

For any input tensor $X \in \mathbb{Z}_\beta^{n_1\times\cdots\times n_d}$ with $rank(X) \leq k$, by Theorem B.3, $eval(\mathcal{F}(\cdot))$ correctly reconstructs it. Also, the product of it with the matrix H matches the sketch v. On the other hand, when the input $rank(X) > k$, by Lemma B.2 under the SIS hardness assumption, there does not exist a tensor Y with $rank(Y) \leq k$ such that $X \neq Y$ and $Hy = v = Hx \mod q$, for x, y being the vectorization of X, Y, respectively. Therefore, in this case Algorithm 6 outputs $None$, as desired.

The random matrix H used in Algorithm 6 can be generated on the fly in the random oracle model. Therefore, the recovery algorithm only stores a sketch vector of length $\tilde{O}(k(n_1+\cdots+n_d))$ with entries bounded by $\mathrm{poly}(n)$. Also, as stated in Remark B.4, the fast recovery scheme $\mathcal{F}(\cdot)$ takes $\tilde{O}(k(n_1+\cdots+n_d))$ bits of space, so the total space usage is $\tilde{O}(k(n_1+\cdots+n_d))$. Both the evaluation of the fast recovery scheme $eval(\mathcal{F}(\cdot))$ and the comparison of vectors take $\mathrm{poly}(n)$ time, giving overall $\mathrm{poly}(n)$ time. □