Time-Space Tradeoffs for Distinguishing Distributions and Applications to Security of Goldreich's PRG

Sumegha Garg
Department of Computer Science, Princeton University, NJ, USA
sumegha.garg@gmail.com

Pravesh K. Kothari[1]
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
kotpravesh@gmail.com

Ran Raz
Department of Computer Science, Princeton University, NJ, USA
ran.raz.mail@gmail.com

Abstract

In this work, we establish lower-bounds against memory bounded algorithms for distinguishing between natural pairs of related distributions from samples that arrive in a streaming setting.

Our first result applies to the problem of distinguishing the uniform distribution on $\{0,1\}^n$ from uniform distribution on some unknown linear subspace of $\{0,1\}^n$. As a specific corollary, we show that any algorithm that distinguishes between uniform distribution on $\{0,1\}^n$ and uniform distribution on an $n/2$-dimensional linear subspace of $\{0,1\}^n$ with non-negligible advantage needs $2^{\Omega(n)}$ samples or $\Omega(n^2)$ memory (tight up to constants in the exponent).

Our second result applies to distinguishing outputs of Goldreich's *local* pseudorandom generator from the uniform distribution on the output domain. Specifically, Goldreich's pseudorandom generator $\mathcal{G}$ fixes a predicate $P : \{0,1\}^k \to \{0,1\}$ and a collection of subsets $S_1, S_2, \ldots, S_m \subseteq [n]$ of size k. For any seed $x \in \{0,1\}^n$, it outputs $P(x_{S_1}), P(x_{S_2}), \ldots, P(x_{S_m})$ where x_{S_i} is the projection of x to the coordinates in S_i. We prove that whenever P is t-resilient (all non-zero Fourier coefficients of $(-1)^P$ are of degree t or higher), then no algorithm, with $< n^\epsilon$ memory, can distinguish the output of $\mathcal{G}$ from the uniform distribution on $\{0,1\}^m$ with a large inverse polynomial advantage, for stretch $m \le \left(\frac{n}{t}\right)^{\frac{(1-\epsilon)}{36}\cdot t}$ (barring some restrictions on k). The lower bound holds in the streaming model where at each time step i, $S_i \subseteq [n]$ is a randomly chosen (ordered) subset of size k and the distinguisher sees either $P(x_{S_i})$ or a uniformly random bit along with S_i.

An important implication of our second result is the security of Goldreich's generator with super linear stretch (*in the streaming model*), against memory-bounded adversaries, whenever the predicate P satisfies the *necessary* condition of t-resiliency identified in various prior works.

Our proof builds on the recently developed machinery for proving time-space trade-offs (Raz 2016 and follow-ups). Our key technical contribution is to adapt this machinery to work for *distinguishing* problems in contrast to prior works on similar results for *search*/learning problems.

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization

Keywords and phrases memory-sample tradeoffs, bounded storage cryptography, Goldreich's local PRG, distinguishing problems, refuting CSPs

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.21

Category RANDOM

[1] Part of this work was completed when Pravesh was at Princeton University and Institute for Advanced Study, Princeton.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020).
Editors: Jarosław Byrka and Raghu Meka; Article No. 21; pp. 21:1–21:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Related Version A full version of this paper is available at `https://arxiv.org/abs/2002.07235`.

Funding *Ran Raz*: Research supported by the Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award and by the National Science Foundation grant No. CCF-1714779.

Acknowledgements We would like to thank Avishay Tal and David Woodruff for the discussions about the problem of distinguishing subspaces.

1 Introduction

This work is motivated by the following basic question: suppose an algorithm is provided with a stream of m i.i.d. samples from a random source. What's the minimum memory required to decide whether the source is "truly random" or "pseudorandom"?

Algorithmically distinguishing perfect randomness from pseudorandomness naturally arises in the context of learning theory (and can even be equivalent to learning in certain models [20, 24, 60, 36]), pseudorandomness and cryptography.

There has been a surge of progress in proving lower bounds for memory-bounded streaming algorithms beginning with Shamir [55] and Steinhardt-Valiant-Wager [57] who conjectured a $\Omega(n^2)$ memory lower bound for learning parity functions with $2^{o(n)}$ samples. This conjecture was proven in [53]. In a follow up work, this was generalized to learning sparse parities in [35] and more general learning problems in [54, 29, 45, 14, 18, 47, 46, 56, 30].

All of these lower bounds hold for learning (more generally, *search*) problems that ask to identify an unknown member of a target function class from samples. In this work, we build on the progress above and develop techniques to show lower bounds for apparently easier task of simply distinguishing uniformly distributed samples from pseudorandom ones. [25] studies the related problem of distribution testing under communication and memory constraints. [25] gave a one-pass streaming algorithm (and a matching lower bound for a broad range of parameters) for uniformity testing on $[N]$ that uses m memory and $O(N \log(N)/(m\epsilon^4))$ samples for distinguishing between uniform distribution on $[N]$ and any distribution that is ϵ-far from uniform.

As we next discuss, our results have consequences of interest in cryptography (ruling out memory-bounded attacks on Goldreich's pseudorandom generator [31] in the streaming model) and average-case complexity (unconditional lower bounds on the number of samples needed, for memory-bounded algorithms, to refute random constraint satisfaction problems, in the streaming model).

1.1 Our Results

We now describe our results in more detail. Our main results show memory-sample trade-offs for distinguishing between truly random and pseudorandom sources for the following two settings:

1. **Uniform vs k-Subspace Source**: The pseudorandom subspace source of dimension k chooses some arbitrary k-dimensional linear subspace $S \subseteq \{0,1\}^n$ and draws points uniformly from S. The truly random source draws points uniformly from $\{0,1\}^n$.
2. **Uniform vs Local Pseudorandom Source**: The pseudorandom source fixes a k-ary Boolean predicate $P : \{0,1\}^k \to \{0,1\}$. It chooses a uniformly random $x \in \{0,1\}^n$ and generates samples $(\alpha, b) \in [n]^{(k)} \times \{0,1\}$ where $[n]^{(k)}$ represents the set of all ordered k-tuples with exactly k elements from $[n]$ and α is chosen uniformly at random from $[n]^{(k)}$ and b is the evaluation of P at x^α - the k-bit string obtained by projecting x onto the coordinates indicated by α. The truly random source generates samples (α, b) where $\alpha \in [n]^{(k)}$ and $b \in \{0,1\}$ are chosen uniformly and independently.

We model our algorithm by a *read-once branching program* (ROBP) of width 2^b (or memory b) and length m. Such a model captures any algorithm that takes as input a stream of m samples and has a memory of at most b bits. Observe that there's no restriction on the computation done at any node of an ROBP.

Roughly speaking, this model gives the algorithm unbounded computational power and bounds only its memory-size and the number of samples used.

Our first main result shows a lower bound on memory-bounded ROBPs for distinguishing between uniform and k-subspace sources.

▶ **Theorem 1** (Uniform vs Subspace Sources). *Any algorithm that distinguishes between uniform and subspace source of dimension k (assuming $k > c \log n$ for some large enough constant c) with probability at least $1/2 + 2^{-o(k)}$ requires either a memory of $\Omega(k^2)$ or at least $2^{\Omega(k)}$ samples. In particular, distinguishing between the uniform distribution on $\{0,1\}^n$ and the uniform distribution on an unkown linear subspace of dimension $n/2$ in $\{0,1\}^n$ requires $\Omega(n^2)$ memory or $2^{\Omega(n)}$ samples.*

Crouch et. al. [16] recently proved that any algorithm that uses at most $n/16$ bits of space requires $\Omega(2^{n/16})$ samples to distinguish between uniform source and a subspace source of dimension $k = n/2$. They suggest the question of improving the space bound to $\Omega(n^2)$ while noting that their techniques do not suffice. For $k = \Theta(n)$, our lower bound shows that any algorithm with memory at most cn^2 for some absolute constant c requires $2^{\Omega(n)}$ samples. This resolves their question.

Upper bound. In Section 4, we exhibit a simple explicit branching program that uses $2^{O(k)}$ samples and $O(1)$ memory to succeed in solving the distinguishing problem with probability $3/4$. We also show a simple algorithm that uses $O(k^2)$ memory and $O(k)$ samples, and succeeds in solving the distinguishing problem with probability $3/4$. *Thus, in the branching program model, the lower bound is tight up to constants in the exponent.*

Our second main result gives a memory-sample trade-off for the uniform vs local pseudorandom source problem for all predicates that have a certain well-studied pseudorandom property studied in cryptography under the name of *resilience.*

A k-ary Boolean function P is said to be *t-resilient* if t is the maximum integer such that $(-1)^P$ (taking the range of the boolean function to be {-1,1}) has zero correlation with every parity function of at most $t-1$ out of k bits. In particular, the parity function on k bits is k-resilient.

▶ **Theorem 2** (Uniform vs Local Pseudorandom Sources). *Let $0 < \epsilon < 1 - 3\frac{\log 24}{\log n}$ and P be a t-resilient k-ary predicate for $k < n^{(1-\epsilon)/6}/3, n/c^2$. Then, any ROBP that succeeds with probability at least $1/2 + \Omega((\frac{t}{n})^{\Omega(t\cdot(1-\epsilon))})$ at distinguishing between uniform and local pseudorandom source for predicate P, requires $(\frac{n}{t})^{\Omega(t\cdot(1-\epsilon))}$ samples or n^ϵ memory.*

Upper bound. In Subsection 5.3, we give an algorithm that takes $(n^\epsilon + k)k \log n$ memory and $(n^{(1-\epsilon)k})(n^\epsilon + k)$ samples, and distinguishes between uniform and local pseudorandom source for any predicate P, with probability 99/100. Thus, the lower bounds are almost tight up to $\log n$ factors and constant factors in the exponent for certain predicates ($t = \Omega(k)$). The question of whether there exists a better algorithm that runs in $O(n^{(1-\epsilon)t})$ samples and $O(n^\epsilon)$ memory, and distinguishes between uniform and local pseudorandom source with high probability, for t-resilient predicates P, remains open.

[2] c is a large enough constant

This result has interesting implications for well-studied algorithmic questions in average-case complexity and cryptography such as refuting random constraint satisfaction [27, 2, 52, 37]) and existence of local pseudorandom generators [17, 49, 11, 43, 4, 5] with super linear stretch where a significant effort has focused on proving lower bounds on various restricted models such as propositional and algebraic proof systems, spectral methods, algebraic methods and semidefinite programming hierarchies. While bounded memory attacks are well-explored in cryptography [44, 15, 10, 9, 59, 26, 53, 61, 32, 58, 34, 19], to the best of our knowledge, memory has not been studied as explicit resource in this context. We discuss these applications further in the paper.

For the special case when $P(x) = \sum_i^k x^i \mod 2$, the parity function on k bits, we can prove stronger results for a wider range of parameters.

▶ **Theorem 3** (Uniform vs Local Pseudorandom Sources with Parity Predicate)**.** *Let $0 < \epsilon < 1 - 3\frac{\log 24}{\log n}$ and P be the parity predicate on k bits for $0 < k < n/c$ (c is a large enough constant). Suppose there's a ROBP that distinguishes between uniform and local pseudorandom source for the parity predicate, with probability at least $1/2 + s$ and uses $< n^\epsilon$ memory. If $s > \Omega\left(\left(\frac{k}{n}\right)^{\Omega((1-\epsilon)\cdot k)}\right)$, then, the ROBP requires $\left(\frac{n}{k}\right)^{(\Omega((1-\epsilon)\cdot k)}$ samples.*

A recent concurrent work [19] builds symmetric encryption schemes which are secure against memory bounded adversaries, where each ciphertext makes at most k calls to a "random" function. Even though the applications and techniques are different, the setting of the analysis is almost identical and the paper obtains better bounds (in terms of constants in the exponent) for the (and only the) parity predicate on k bits.

The above results show lower bounds for *sublinear* memory algorithms. For a slight variant of the above uniform vs local pseudorandom source problem, we can in fact upgrade our results to obtain the following lower bounds against *super-linear* memory algorithms. See Section 4 for details.

▶ **Theorem 4.** *For large enough n and $k > c\log n$ (where c is a large enough constant) and $k \leq \frac{n}{4}$, any algorithm that can distinguish satisfiable sparse parities of sparsity k on n variables (of type $(a, b) = (a^1, a^2, ..., a^n, b) \in \{0, 1\}^{n+1}$, where $\forall i \in [n],\ a^i = 1$ with probability $\frac{k}{n}$ and $b = \langle a, x\rangle$) from random ones (of similar type (a, b) but b is now chosen uniformly at random from $\{0, 1\}$), with success probability at least $\frac{1}{2} + 2^{-o(k)}$, requires either a memory of size $\Omega(nk)$ or $2^{\Omega(k)}$ samples.*

In Remark 15, we observe that the above theorem is almost tight. Specifically, we observe that there are ROBPs that use a constant memory and $O(n2^{O(k)})$ samples or $O(nklogn)$ memory and $O(n)$ samples to distinguish uniform sources from locally pseudorandom ones.

1.2 Applications to Security of Goldreich's Pseudorandom Generator

A fundamental goal in cryptography is to produce secure constructions of cryptographic primities that are highly efficient. In line with this goal, Goldreich [31] proposed a candidate one-way function given by the following pseudorandom mapping that takes n-bit input x and outputs m bits: fix a *predicate* $P : \{0, 1\}^k \to \{0, 1\}$, pick $a_1, a_2, \ldots, a_m$ uniformly at random[3] from $[n]^{(k)}$ and output $P(x^{a_1}), P(x^{a_2}), \ldots, P(x^{a_m})$. Here, $a_1, \ldots, a_m$ and P are public and the seed x is secret. Later works (starting with [48]) suggested using this candidate as pseudorandom generator.

[3] More generally, Goldreich proposed that $a_1, a_2, \ldots, a_m$ could be chosen in a pseudorandom way so as to ensure a certain "expansion" property. We omit a detailed discussion here.

The main question of interest is the precise trade-off between the *locality* k and the *stretch* m for a suitable choice of the predicate P. In several applications, we need that the generator has a super-linear stretch (i.e. $m = n^{1+\delta}$ for some $\delta > 0$) with constant locality (i.e. $k = O(1)$).

The simplicity and efficiency of such a candidate is of obvious appeal. This simplicity has been exploited to yield a host of applications including public-key cryptography from combinatorial assumptions [6], highly efficient secure multiparty computation [33] and most recently, basing *indistinguishability obfuscation* on milder assumptions [38, 3, 41, 39, 40].

Evidence for the security of Goldreich's candidate has been based on analyzing natural classes of attacks based on propositional proof systems [1], spectral methods and semidefinite programming hierarchies [51, 2, 12, 37, 42, 11] and algebraic methods [7, 8]. In particular, previous works [37, 8] identified t-resiliency of the predicate P as a necessary condition for the security of the candidate for $m = n^{\Omega(t)}$ stretch.

The *uniform vs local pseudorandom source* problem considered in this work is easily seen as the algorithmic question of distinguishing the output stream generated by Goldreich's candidate generator from a uniformly random sequence of bits. In particular, our results imply security of Goldreich's candidate against bounded memory algorithms for super-linear stretch when instantiated with any t-resilient predicate for large enough constant t (but in the streaming model). Goldreich's candidate generator would fix the sets $a_1, a_2, \ldots, a_m$ (which are public) and output $P(x^{a_1}), P(x^{a_2}), \ldots, P(x^{a_m})$ for n sized input x $(m > n)$. We prove the security of Goldreich's generator in the model where $a_1, a_2, \ldots, a_m$, still public, are chosen uniformly at random from $[n]^{(k)}$ and streamed with the generated bits.

We note that our lower bounds continue to hold even when the locality k grows polynomially with the seed length n.

▶ **Corollary 5** (Corollary of Theorem 2). *Let $0 < \epsilon < 1 - 3\frac{\log 24}{\log n}$ and P be a t-resilient k-ary predicate for $k = O(n^{(1-\epsilon)/6})$. Then, Goldreich's PRG, when instantiated with any t-resilient k-ary predicate P such that $k \geq t > 36$ and stretch $m = (n/t)^{O(t)(1-\epsilon)}$, is secure against all read-once branching programs with memory-size bounded from above by n^{ϵ}, in the streaming model.*

1.3 Applications to Refuting Random CSPs

Theorems 2 and 3 can also be intepreted as lower bounds for the problem of refuting random constraint satisfaction problems.

A random CSP with predicate $P : \{0,1\}^k \to \{0,1\}$ is a constraint satisfaction problem on n variables $x \in \{0,1\}^n$. More relevant to us is the variant where the constraints are randomly generated as follows: choose an ordered k-tuple of variables a from $[n]$ at random, a bit $b \in \{0,1\}$ at random and impose a constraint $P(x^a) = b$. When the number of constraints $m \gg n$, the resulting instance is unsatisfiable with high probability for any non-constant predicate P. The natural algorithmic problem in this regime is that of *refutation* - efficiently finding a short witness that certifies that the given instance is far from satisfiable. It is then easy to note that the uniform vs local pseudorandom source problem is the task of distinguishing between constraints in a random CSP (with predicate P) and one with a satisfying assignment. Note that refutation is formally harder than the task of distinguishing between a random CSP and one that has a satisfying assignment.

Starting with investigating in proof complexity, random CSPs have been intensively studied in the past three decades. When P is t-resilient for $t \geq 3$, all known efficient algorithms [2] require $m \gg n^{1.5}$ samples for the refutation problem. This issue was brought to the forefront in [27] where Feige made the famous "Feige's Hypothesis" conjecturing the impossibility of refuting random 3SAT in polynomial time with $\Theta(n)$ samples. Variants of Feige's hypothesis for other predicates have been used to derive hardness results in both supervised [21, 22, 23] and unsupervised machine learning [13].

In [51], t-resilience was noted as a necessary condition for the refutation problem to be hard. Our Theorems 2 and 3 confirm this as a sufficient condition for showing lower-bounds for the refutation (in fact, even for the easier "distinguishing" variant) of random CSPs, with t-resilient predicates, in the streaming model with bounded memory.

2 Preliminaries

Denote by log the logarithm to base 2. We use $Ber(p)$ to denote the Bernoulli distribution with parameter p (probability of being 1). We use $[n]$ to denote the set $\{1, 2, ..., n\}$.

For a random variable Z and an event E, we denote by $\mathbb{P}_Z$ the distribution of the random variables Z, and we denote by $\mathbb{P}_{Z|E}$ the distribution of the random variable Z conditioned on the event E.

Given an $n-$bit vector $y \in \{0,1\}^n$, we use y^i to denote the i^{th} coordinate of y, that is, $y = (y^1, y^2, ..., y^n)$. We use $y^{-i} \in \{0,1\}^{n-1}$ to denote y but with the ith coordinate deleted. Given two $n-$bit vectors y, y', we use $\langle y, y' \rangle$ to denote the inner product of y and y' modulo 2, that is, $\langle y, y' \rangle = \sum_{i=1}^{n} y^i y'^i \mod 2$. We use $|y|$ to denote the number of ones in the vector y.

Given a set S, we use $y \in_R S$ to denote the random process of picking y uniformly at random from the set S. Given a probability distribution D, we use $y \sim D$ to denote the random process of sampling y according to the distribution D.

Next, we restate (for convenience) the definitions and results from previous papers [53, 54, 35, 29] that we use.

Viewing a Learning Problem as a Matrix

Let X, A be two finite sets of size larger than 1.

Let $M : A \times X \to \{-1, 1\}$ be a matrix. The matrix M corresponds to the following learning problem: There is an unknown element $x \in X$ that was chosen uniformly at random. A learner tries to learn x from samples (a, b), where $a \in A$ is chosen uniformly at random and $b = M(a, x)$. That is, the learning algorithm is given a stream of samples, $(a_1, b_1), (a_2, b_2) \ldots$, where each a_t is uniformly distributed and for every t, $b_t = M(a_t, x)$.

These papers model the learner for the learning problem corresponding to the matrix M using a branching program:

▶ **Definition 6** (Branching Program for a Learning Problem). *A branching program of length m and width d, for learning, is a directed (multi) graph with vertices arranged in $m+1$ layers containing at most d vertices each. In the first layer, that we think of as layer 0, there is only one vertex, called the start vertex. A vertex of outdegree 0 is called a leaf. All vertices in the last layer are leaves (but there may be additional leaves). Every non-leaf vertex in the program has $2|A|$ outgoing edges, labeled by elements $(a, b) \in A \times \{-1, 1\}$, with exactly one edge labeled by each such (a, b), and all these edges going into vertices in the next layer. Each leaf v in the program is labeled by an element $\tilde{x}(v) \in X$, that we think of as the output of the program on that leaf.*

Computation-Path: *The samples $(a_1, b_1), \ldots, (a_m, b_m) \in A \times \{-1, 1\}$ that are given as input, define a computation-path in the branching program, by starting from the start vertex and following at step t the edge labeled by (a_t, b_t), until reaching a leaf. The program outputs the label $\tilde{x}(v)$ of the leaf v reached by the computation-path.*

Success Probability: *The success probability of the program is the probability that $\tilde{x} = x$, where $\tilde{x}$ is the element that the program outputs, and the probability is over $x, a_1, \ldots, a_m$ (where x is uniformly distributed over X and $a_1, \ldots, a_m$ are uniformly distributed over A, and for every t, $b_t = M(a_t, x)$).*

▶ **Theorem 7** ([53, 54, 29])**.** *Any branching program that learns $x \in \{0,1\}^n$, from random linear equations over $\mathbb{F}_2$ with success probability 2^{-cn} requires either a width of $2^{\Omega(n^2)}$ or a length of $2^{\Omega(n)}$ (where c is a small enough constant).*

▶ **Theorem 8** ([29])**.** *Any branching program that learns $x \in \{0,1\}^n$, from random sparse linear equations, of sparsity exactly ℓ, over $\mathbb{F}_2$ with success probability 2^{-cl} (where c is a small enough constant) requires:*

1. *Assuming $\ell \leq n/2$: either a width of size $2^{\Omega(n \cdot \ell)}$ or length of $2^{\Omega(\ell)}$.*
2. *Assuming $\ell \leq n^{0.9}$: either a width of size $\Omega(n \cdot \ell^{0.99})$ or length of $\ell^{\Omega(\ell)}$.*

Norms and Inner Products

Let $p \geq 1$. For a function $f : X \to \mathbb{R}$, denote by $\|f\|_p$ the ℓ_p norm of f, with respect to the uniform distribution over X, that is:

$$\|f\|_p = \left(\mathop{\mathbf{E}}_{x \in_R X} [|f(x)|^p] \right)^{1/p} .$$

For two functions $f, g : X \to \mathbb{R}$, define their inner product with respect to the uniform distribution over X as

$$\langle f, g \rangle = \mathop{\mathbf{E}}_{x \in_R X} [f(x) \cdot g(x)].$$

For a matrix $M : A \times X \to \mathbb{R}$ and a row $a \in A$, we denote by $M_a : X \to \mathbb{R}$ the function corresponding to the a-th row of M. Note that for a function $f : X \to \mathbb{R}$, we have $\langle M_a, f \rangle = \frac{(M \cdot f)_a}{|X|}$.

▶ **Definition 9** (L_2-Extractor, [29])**.** *Let X, A be two finite sets. A matrix $M : A \times X \to \{-1, 1\}$ is a (k', ℓ')-L_2-Extractor with error $2^{-r'}$, if for every non-negative $f : X \to \mathbb{R}$ with $\frac{\|f\|_2}{\|f\|_1} \leq 2^{\ell'}$ there are at most $2^{-k'} \cdot |A|$ rows a in A with*

$$\frac{|\langle M_a, f \rangle|}{\|f\|_1} \geq 2^{-r'} .$$

▶ **Lemma 10** ([35])**.** *Let T_l be the set of n-bit vectors with sparsity exactly-l for $l \in \mathbb{N}$, that is, $T_l = \{x \in \{0,1\}^n \mid \sum_{i=1}^n x^i = l\}$. Let $\delta \in (0, 1]$. Let $\mathcal{B}_{T_l}(\delta) = \{\alpha \in \{0,1\}^n \mid \left|\boldsymbol{E}_{x \in T_l} (-1)^{\langle \alpha, x \rangle}\right| > \delta\}$. Then, for $\delta \geq (\frac{8l}{n})^{\frac{l}{2}}$,*

$$|\mathcal{B}_{T_l}(\delta)| \leq 2e^{-\delta^{2/l} \cdot n/8} \cdot 2^n$$

Branching Program for a Distinguishing Problem

Let X, A be two finite sets of size larger than 1. Let D_0 be a distribution over the sample space $|A|$. Let $\{D_1(x)\}_{x \in X}$ be a set of distributions over the sample space $|A|$. Consider the following distinguishing problem: An unknown $\mathbf{b} \in \{0, 1\}$ is chosen uniformly at random. If $\mathbf{b} = 0$, the distinguisher is given independent samples from D_0. If $\mathbf{b} = 1$, an unknown $x \in X$ is chosen uniformly at random, and the distinguisher is given independent samples from $D_1(x)$. The distinguisher tries to learn $\mathbf{b}$ from the samples drawn according to the respective distributions.

Formally, we model the distinguisher by a branching program as follows.

▶ **Definition 11** (Branching Program for a Distinguishing Problem). *A branching program of length m and width d, for distinguishing, is a directed (multi) graph with vertices arranged in $m+1$ layers containing at most d vertices each. In the first layer, that we think of as layer 0, there is only one vertex, called the start vertex. A vertex of outdegree 0 is called a leaf. All vertices in the last layer are leaves (but there may be additional leaves). Every non-leaf vertex in the program has $|A|$ outgoing edges, labeled by elements $a \in A$, with exactly one edge labeled by each such a, and all these edges going into vertices in the next layer. Each leaf v in the program is labeled by a $\tilde{\boldsymbol{b}}(v) \in \{0,1\}$, that we think of as the output of the program on that leaf.*

Computation-Path: *The samples $a_1, \ldots, a_m \in A$ that are given as input, define a computation-path in the branching program, by starting from the start vertex and following at step t the edge labeled by a_t, until reaching a leaf. The program outputs the label $\tilde{\boldsymbol{b}}(v)$ of the leaf v reached by the computation-path.*

Success Probability: *The success probability of the program is the probability that $\tilde{\boldsymbol{b}} = \boldsymbol{b}$, where $\tilde{\boldsymbol{b}}$ is the element that the program outputs, and the probability is over $\boldsymbol{b}, x, a_1, \ldots, a_m$ (where $\boldsymbol{b}$ is uniformly distributed over $\{0,1\}$, x is uniformly distributed over X and $a_1, \ldots, a_m$ are independently drawn from D_0 if $\boldsymbol{b} = 0$ and $D_1(x)$ if $\boldsymbol{b} = 1$).*

3 Overview of the Proofs

We prove our theorems using two different techniques. We prove Theorems 1 and 4 through reductions to the memory-sample lower bounds for the corresponding learning problems in Section 4. Informally, for Theorem 1, we construct a branching program that learns the unknown vector x from random linear equations in $\mathbb{F}_2$ by guessing each bit one by one sequentially and using the distinguisher, for distinguishing subspaces from uniform, to check if it guessed correctly. Then, we are able to lift the previously-known memory-sample lower bounds for the learning problem (Theorem 7) to the distinguishing problem. Similarly, we lift the memory-sample lower bounds for a variant of the learning problem in Theorem 8 to the get Theorem 4.

Theorems 2 and 3 are restated in Section 5. A brief overview of these theorems are as follows. Recall, a pseudorandom source fixes a k-ary Boolean predicate $P : \{0,1\}^k \to \{0,1\}$. It chooses a uniformly random $x \in \{0,1\}^n$ and generates samples $(\alpha, b) \in [n]^{(k)} \times \{0,1\}$ where α is a uniformly random (ordered) k-tuple of indices in $[n]$ and b is the evaluation of P at x^α - the k-bit string obtained by projecting x onto the coordinates indicated by α. The truly random source samples (α, b) where $\alpha \in [n]^{(k)}$ and $b \in \{0,1\}$ are chosen uniformly and independently. The problem for a distinguisher is to correctly guess whether the m samples are generated by a pseudorandom or a uniform source, when the samples arrive in a stream. We first show through a *hybrid argument* that a distinguisher A that distinguishes between the uniform and pseudorandom source, with an advantage of s over $1/2$, can also distinguish (with advantage of at least s/m) when only the jth (for some j) sample is drawn from the "unknown source", the first $j-1$ samples are drawn from the pseudorandom source and the last $m-j$ samples are drawn from the uniform source.

Let v be the memory state of A after seeing the first $j-1$ samples, which were generated from a pseudorandom source with a seed x picked uniformly at random from $\{0,1\}^n$. Let $\mathbb{P}_{x|v}$ be the probability distribution of the random variable x conditioned on reaching v. If the jth sample is generated using the same pseudorandom source, then $\forall \alpha \in [n]^{(k)}$, the bit b is 0 with probability $\sum_{x':P(x'^\alpha)=0} \mathbb{P}_{x|v}(x')$ and 1 with probability $1 - \sum_{x':P(x'^\alpha)=0} \mathbb{P}_{x|v}(x')$. If the jth sample is generated using the uniform source, then $\forall \alpha \in [n]^{(k)}$, the bit b is 0 with probability 1/2 and 1 with probability 1/2. Thus, for any α, A can identify the "unknown source" with an at most $\left|\sum_{x':P(x'^{\alpha})=0} \mathbb{P}_{x|v}(x') - 1/2\right|$ advantage. We show that when A has low memory ($< n^{\epsilon}$ for some $0 < \epsilon < 1$), then with high probability, it reaches a state v such that $\mathbb{P}_{x|v}$ has high min-entropy (informally, it's hard to determine the seed for the pseudorandom source). The argument till now (last 2 paragraphs) is common and has been previously used to prove that a "source" fools bounded space, such as in [50].

Next, we use t-resiliency of P to show that when $\mathbb{P}_{x|v}$ has high min-entropy, then with high probability over $\alpha \in [n]^{(k)}$, b behaves almost like in a uniform source (Lemma 19), that is, $|\sum_{x':P(x'^{\alpha})=0} \mathbb{P}_{x|v}(x') - 1/2|$ is small. Hence, with high probability, it's hard for A to judge with 'good' advantage whether b was generated from a pseudorandom or a uniform source. Note that the last $m - j$ samples generated by a uniform source can't better this advantage.

4 Time-Space Tradeoff through Reduction to Learning

In this section, we will state time-space tradeoffs for the following distinguishing problems, which are proved using black-box reduction from the corresponding learning problems.

Distinguishing Subspaces from Uniform

Informally, we study the problem of distinguishing between the cases when the samples are drawn from a uniform distribution over $\{0,1\}^n$ and when the samples are drawn randomly from a subspace of rank k over $\mathbb{F}_2$. Let $L(k,n)$ be the set of all linear subspaces of dimension k ($\subseteq \{0,1\}^n$), that is, $L(k,n)$ contains all subspaces V such that $V = \{v \in \{0,1\}^n \mid \langle w_i, v\rangle = 0 \ \forall i \in [n-k]\}$ for some linearly independent vectors $w_1, w_2, ..., w_{n-k}$. Formally, we consider distinguishers for distinguishing between the following distributions:

1. D_0: Uniform distribution over $\{0,1\}^n$.
2. $D_1(S)$, $S \in L(k,n)$: Uniform distribution over S.

Note: If the subspace S is revealed, it's easy for a branching program of constant width to distinguish w.h.p. by checking the inner product of the samples with a vector in the orthogonal complement of S.

A distinguisher can distinguish subspaces if for an unknown random linear subspace $S \in L(k,n)$, it can distinguish between D_0 and $D_1(S)$. Formally, a distinguisher L, after seeing m samples, has a success probability of p if

$$\frac{\Pr_{u_1,...,u_m \sim D_0}[L(u_1,...,u_m) = 0] + \Pr_{S \in_R L(k,n); u_1,...,u_m \sim D_1(S)}[L(u_1,...,u_m) = 1]}{2} = p \quad (1)$$

▶ **Theorem 12.** *For $k > c_2 \log n$ (where c_2 is a large enough constant), any algorithm that can distinguish k-dimensional subspaces over $\mathbb{F}_2^n$ from $\mathbb{F}_2^n$ ($\{0,1\}^n$), when samples are drawn uniformly at random from the subspace or $\mathbb{F}_2^n$ respectively, with success probability at least $\frac{1}{2} + 2^{-o(k)}$ requires either a memory of size $\Omega(k^2)$ or $2^{\Omega(k)}$ samples.*

For the proof, refer to the full version of the paper [28]. Briefly, we prove that using a distinguisher for distinguishing subspaces, we can construct a branching program that learns an unknown bit vector x from random linear equations over $\mathbb{F}_2$. Then, we are able to lift the time-space tradeoffs of Theorem 7.

▶ Remark 13 (Tightness of the Lower Bound). We note two easy upper bounds that show that our results in Theorem 12 are tight (up to constants in the exponent). Firstly, we observe an algorithm B_1 that distinguishes subspaces of dimension k from uniform, using $O(k^2)$ memory and $O(k)$ samples, with probability at least 3/4 ($0 < k \leq n-1$). B_1 stores the first $\min(8k, n)$ bits of the first $8k$ samples (in $O(k^2)$ memory); outputs 1 if the samples (projected onto the first $\min(8k, n)$ coordinates) belong to a $\leq k$-dimensional subspace (of $\{0,1\}^{\min(8k,n)}$), and 0 otherwise (can be checked using gaussian elimination). When the samples are drawn from $D_1(S)$ for some k-dimensional subspace S, then B_1 always outputs the correct answer. When the samples are drawn from a uniform distribution on $\{0,1\}^n$, the probability that $8k$ samples form a k-dimensional subspace is at most

$$\binom{8k}{k} \cdot \frac{1}{2^{7k}} \leq (8e)^k 2^{-7k} < 2^{-2k} \leq 1/4$$

(because, if the $8k$ samples form a k-dimensional subspace, then at least $7k$ of them are linearly dependent on the previously stored samples and that happens with at most $1/2$ probability for each sample). Hence, B_1 errs with at most $1/4$ probability.

Secondly, we observe that there exists a branching program that distinguishes subspaces of dimension k from uniform using constant width and $O(k \cdot 2^k)$ length with probability at least $3/4$. Before, we show a randomized algorithm P that distinguishes between D_0 and $D_1(S)$ for every $S \in L(k, n)$ with high probability. P is described as follows:

1. Repeat steps 2 to 3 sequentially for $t = 10 \cdot 2^k$ iterations.
2. Pick a non-zero vector v uniformly at random from $\{0,1\}^n$. For the next $2k$ samples (of the form $a \in \{0,1\}^n$), check if $\langle a, v\rangle = 0$.
3. If all the $2k$ samples are orthogonal to v, exit the loop and output 1.
4. Output 0 (None of the randomly chosen vectors were orthogonal to all the samples seen in its corresponding iteration).

The number of samples seen by P is $20k \cdot 2^k$. Now, we prove that for every subspace S of dimension k, that is, $S \in L(k, n)$, P distinguishes between D_0 and $D_1(S)$ with probability at least $1 - \frac{1}{2}(e^{-5} + \frac{10}{2^k}) \geq 3/4$[4].

When the samples are drawn from D_0, the probability that P outputs 1 is equal to the probability that in at least one of the t iterations, the randomly chosen non-zero vector v was orthogonal to the $2k$ samples drawn uniformly from $\{0,1\}^n$. Here, the probability is over v and the samples. By union bound, we can bound the probability of outputting 1 (error) by

$$10 \cdot 2^k \cdot \left(\frac{1}{2}\right)^{2k} = \frac{10}{2^k}.$$

For a fixed subspace $S \in L(k, n)$, the probability that we pick a non-zero vector $v \in \{0,1\}^n$ that is orthogonal to S is at least $\frac{2^{n-k}-1}{2^n-1} > 2^{-(k+1)}$. Therefore, when the samples are drawn from $D_1(S)$, the probability that P outputs 0 (error) is upper bounded by $\left(1 - \frac{1}{2^{k+1}}\right)^{10 \cdot 2^k} \leq e^{-5}$. Here, the probability is over the vectors v and the samples. Now to construct a constant width but $20k \cdot 2^k$ length branching program that distinguishes with probability at least $3/4$, we consider a bunch of branching programs each indexed by t vectors that are used in step 2 of the algorithm P. It's easy to see that for a fixed set of t vectors, P can be implemented by a constant width branching program. As, when the t vectors are uniformly distributed over $\{0,1\}^n$ (non-zero), P can distinguish with probability at least $3/4$ for every subspace $S \in L(k, n)$, there exists a fixing to the t vectors such that the corresponding branching program distinguishes between D_0 and $D_1(S)$ (when S is chosen uniformly at random from $L(k, n)$) with probability at least $3/4$.

[4] $k \geq 5$

Distinguishing Satisfiable Sparse Equations from Uniform

Informally, we study the problem of distinguishing between the cases when the samples are drawn from satisfiable sparse equations over $\mathbb{F}_2$ and when the samples are drawn from random sparse equations.

Formally, we consider the distinguishing problem between the following two distributions:

1. D_0: Distribution on $(n+1)$-length vectors $(v^1, v^2, ..., v^n, b) \in \{0,1\}^{n+1}$ where $\forall i \in [n]$, v^i is 1 with probability $\frac{k}{n}$ and 0 otherwise, and b is 1 with probability $\frac{1}{2}$ and 0 otherwise.
2. $D_1(x)$, $x \in \{0,1\}^n$: Distribution on $(n+1)$-length vectors $(v^1, v^2, ..., v^n, b) \in \{0,1\}^{n+1}$ where $\forall i \in [n]$, v^i is 1 with probability $\frac{k}{n}$ and 0 otherwise, and $b = \langle v, x\rangle$ where $v = (v^1, v^2, ..., v^n)$.

Here, k is the sparsity parameter.

We say that a distinguisher can distinguish satisfiable sparse equations of sparsity k from random ones if, when x is unknown and chosen uniformly at random from $\{0,1\}^n$, it can distinguish between D_0 and $D_1(x)$. Formally, a distinguisher L, after seeing m, has a success probability of p if

$$\frac{\Pr_{u_1,...,u_m \sim D_0}[L(u_1,...,u_m) = 0] + \Pr_{x \in_R \{0,1\}^n; u_1,...,u_m \sim D_1(x)}[L(u_1,...,u_m) = 1]}{2} \geq p \quad (2)$$

▶ **Theorem 14.** *For large enough n and $k > c_5 \log n$ (where c_5 is a large enough constant) and $k \leq \frac{n}{4}$, any algorithm that can distinguish random sparse parities of sparsity k on n variables from satisfiable ones, with success probability at least $\frac{1}{2} + 2^{-o(k)}$, requires either a memory of size $\Omega(nk)$ or $2^{\Omega(k)}$ samples.*

▶ Remark 15 (Tightness of our Lower Bound). We note two easy upper bounds that show that our results in Theorem 14 are almost tight. Firstly, we observe that there's an algorithm B_1 of memory $O(nk \log n)$ that uses $O(n)$ samples and can distinguish random sparse parities of sparsity k from satisfiable ones, with probability of at least 3/4. B_1 just stores $O(n)$ samples (in $O(nk \log n)$ memory); if there exists x that satisfies all the samples, it outputs 1, otherwise it outputs 0. When the samples are satisfiable, that is, drawn from $D_1(x)$ (for some x), B_1 always outputs 1. When the samples are random, using the union bound, it's easy to see that the probability that there exists an x that satisfies all the $O(n)$ samples is exponentially small in n.

Second, there's an algorithm B_2 of constant memory that uses $O(n \cdot 2^{O(k)})$ samples and can distinguish random sparse parities of sparsity k from satisfiable ones, with probability of at least 3/4. The probability that a learning algorithm sees sample (a, b), such that $a = (1, 0, ..., 0)$, is at least $\frac{ke^{-2k}}{n}$ for $k < n/2$; thus, one can just wait for say 5 such samples and see if the values of b are drawn randomly or are fixed, giving a constant memory and $O(ne^{2k})$ samples algorithm that distinguishes with high probability.

Refer to the full version of the paper [28] for the proof of Theorem 14. Briefly, we prove that using such a distinguisher, we can construct a branching program that learns an unknown bit vector x from sparse linear equations of sparsity k over $\mathbb{F}_2$. Unlike before, when we were able to lift, we are not able to directly lift the time-space tradeoffs of Theorem 8, because these lower bounds hold when the equations are of sparsity *exactly-k*. Following the proof of Theorem 8 in [29] very closely, we can prove the following lemma:

▶ **Lemma 16.** *Any branching program that learns $x \in \{0,1\}^n$ from random linear equations over $\mathbb{F}_2$ of type $(a, b) = (a^1, a^2, ..., a^n, b) \in \{0,1\}^{n+1}$, where $\forall i \in [n]$, $a^i = 1$ with probability $\frac{k}{n}$ and $b = \langle a, x\rangle$, with success probability 2^{-ck}, requires either width of size $2^{\Omega(n \cdot k)}$ or length of $2^{\Omega(k)}$ (where c is a small enough constant, $k \leq \frac{n}{4}$).*

The proof is given in full version [28]. Now through reduction, we are able to lift the time-space tradeoffs of Lemma 16 to get Theorem 14.

5 Sample-Memory Tradeoffs for Resilient Local PRGs

In this section, we prove our lower bound against memory bounded algorithms for distinguishing between streaming outputs of Goldreich's pseudorandom generator and perfectly random bits. Before stating our result in detail, we set up some notation and definitions that will be convenient for us in this section.

5.1 Formal Setup

A k-ary *predicate* P is a Boolean function $P : \{0,1\}^k \to \{0,1\}$. Let $\sum_{\alpha \subseteq [k]} \hat{P}(\alpha)\chi_\alpha$ be the Fourier polynomial for $(-1)^P$ ($(-1)^P(x) = (-1)^{P(x)}$). P is said to be *t-resilient* if t is the maximum positive integer such that $\hat{P}(\alpha) = 0$ whenever $|\alpha| < t$. In particular, the parity function $\langle \alpha, x\rangle$ is $|\alpha|$-resilient. Here, $\chi_\alpha : \{0,1\}^k \to \{-1,1\}$ is such that $\chi_\alpha(x) = (-1)^{\langle \alpha, x\rangle}$.

Let $[n]^{(k)}$ denote the set of all ordered k-tuples of exactly k elements of $[n]$. That is, no element of $[n]$ occurs more than once in any tuple of $[n]^{(k)}$. For any $a \in [n]^{(k)}$, let $a^i \in [n]$ denote the element of $[n]$ appearing in the ith position in a. Given $x \in \{0,1\}^n$ and $a \in [n]^{(k)}$, let $x^a \in \{0,1\}^k$ be defined so that $(x^a)^i = x^{a^i}$ for every $1 \leq i \leq k$.

For any k-ary predicate P, consider the problem of distinguishing between the following two distributions on $(a, b) \in [n]^{(k)} \times \{0,1\}$ where (a, b) are sampled as follows:

1. D_{null}: 1) Choose a uniformly at random from $[n]^{(k)}$, and 2) choose b uniformly at random and independently from $\{0,1\}$.
2. $D_{planted}(x)$, $x \in \{0,1\}^n$: 1) Choose a uniformly at random from $[n]^{(k)}$, and 2) set $b = P(x^a)$.

Note that a is chosen uniformly at random from $[n]^{(k)}$ in both distributions. However, while the bit b is independent of a in D_{null}, it may be correlated with a in $D_{planted}$.

A distinguisher for the above problem gets access to m i.i.d. samples $u_t = (a_t, b_t), t \in [m]$ from one of D_{null} and $D_{planted}(x)$ for a uniformly randomly chosen $x \in \{0,1\}^n$ and outputs either "planted" or "null". We say that the distinguisher succeeds with probability p if

$$\Pr_{u_1,\ldots,u_m \sim D_{null}}[L(u_1, \ldots, u_m) = \text{"null"}] + \Pr_{\substack{x \in_R \{0,1\}^n; \\ u_1,\ldots,u_m \sim D_{planted}(x)}}[L(u_1, \ldots, u_m) = \text{"planted"}]$$

is greater than $2p$. In the language used in the previous sections, think of "null" as being equivalent to 0 and "planted" being equivalent to 1, that is, $D_{null} \equiv D_0$ and $D_{planted}(x) \equiv D_1(x)$. Therefore, the success probability of the distinguisher L can be written as

$$\frac{\Pr_{u_1,\ldots,u_m \sim D_0}[L(u_1, \ldots, u_m) = 0] + \Pr_{x \in_R \{0,1\}^n; u_1,\ldots,u_m \sim D_1(x)}[L(u_1, \ldots, u_m) = 1]}{2} \geq p \quad (3)$$

In particular, if $x \in \{0,1\}^n$ is "revealed" to a distinguishing algorithm, then it is easy to use $\Theta(\log(1/\epsilon))$ samples and constant width branching program to distinguish correctly with probability at least $1 - \epsilon$ between D_{null} and $D_{planted}$.

5.2 Main Result

The main result of this section is the following sample-memory trade-off for any distinguisher.

▶ **Theorem 17.** *Let P be a t-resilient k-ary predicate. Let $0 < \epsilon < 1 - 3\frac{\log 24}{\log n}$ and $k < n/c$. Suppose there's an algorithm that distinguishes between D_{null} and $D_{planted}$ with probability at least $1/2 + s$ and uses $< n^\epsilon$ memory. Then, whenever $0 < t \leq k < n^{\frac{(1-\epsilon)}{6}}/3$ and $s > c_1(\frac{n}{t})^{-(\frac{1-\epsilon}{36})t}$, the algorithm requires $(\frac{n}{t})^{(\frac{1-\epsilon}{36})t}$ samples. Here, c and c_1 are large enough constants.*

Note that when k is a constant, this theorem gives a sample-memory tradeoff even for $\Omega(n)$ memory.

Our argument yields a slightly better quantitative lower bound for the special case when P is the parity function, that is, $P(x) = \left(\sum_{i=1}^{k} x^i\right) \mod 2$. We will represent this function by Xor.

▶ **Theorem 18.** *Let $0 < \epsilon < 1 - 3\frac{\log 24}{\log n}$ and P be the parity predicate Xor on $k = t$ bits. Suppose there's an algorithm that distinguishes between D_{null} and $D_{planted}$ with probability at least $1/2 + s$ and uses $< n^{\epsilon}$ memory. Then for $k \leq n/c^5$, if $s > 3(\frac{n}{k})^{-(\frac{1-\epsilon}{18})k}$, the algorithm requires $(\frac{n}{k})^{(\frac{1-\epsilon}{18})k}$ samples.*

We prove both Theorem 17 and 18 via the same sequence of steps except for a certain quantitative bound presented in Lemma 19. In the next subsection, we give an algorithm that takes $\widetilde{O}(n^{\epsilon} + k)k$ memory and $\widetilde{O}(n^{(1-\epsilon)k})$ samples, and distinguishes between D_{null} and $D_{planted}$ for any predicate P, with probability $99/100$. Thus, the lower bounds are almost tight up to constant factors in the exponent for the parity predicate. The question of whether there exists an algorithm that runs in $O(n^{(1-\epsilon)t})$ samples and $O(n^{\epsilon})$ memory, and distinguishes between D_{null} and $D_{planted}$ with high probability, for t-resilient predicates P, remains open.

5.3 Tightness of the Lower Bound

In this section, we observe that there exists an algorithm A that takes $O((n^{\epsilon} + k) \cdot k \log n)$ memory and $O(n^{(1-\epsilon)k} \cdot (n^{\epsilon} + k))$ samples, and distinguishes between D_{null} and $D_{planted}$ for any predicate P, with probability $99/100$ (for $n^{\epsilon} > 10$).

A runs over $4n^{(1-\epsilon)k} \cdot (n^{\epsilon} + k)$ samples and stores the first $2(n^{\epsilon} + k)$ samples $(a, b) \in [n]^{(k)} \times \{0, 1\}$ such that $a^i \leq n^{\epsilon} + k, \forall i \in [k]$, that is, the bit b depends only on the first $n^{\epsilon} + k$ bits of x under the distribution $D_{planted}(x)$. If A encounters less than $2(n^{\epsilon} + k)$ samples of the above mentioned form, A outputs 1 ("planted"). Otherwise, A goes over all the possibilities of the first $n^{\epsilon} + k$ bits of x ($2^{n^{\epsilon}+k}$ possibilities in total) and checks if it could have generated the stored samples. If there exists a $y \in \{0, 1\}^{n^{\epsilon}+k}$ that generated the stored samples, A outputs 1 ("planted"), otherwise A outputs 0.

It's easy to see that A uses $m = 4n^{(1-\epsilon)k} \cdot (n^{\epsilon} + k)$ samples and at most $2(n^{\epsilon} + k) \cdot k \log n$ memory (as it takes only $k \log n$ memory to store a sample). Next, we calculate the probability of success. Let Z_j be a random variable as follows: $Z_j = 1$ if the j^{th} sample (a_j, b_j) is such that $a_j^i \leq n^{\epsilon} + k, \forall i \in [k]$ and 0 otherwise.

$$\Pr[Z_j = 1] = \frac{|[n^{\epsilon} + k]^{(k)}|}{|[n]^{(k)}|} \geq n^{-(1-\epsilon)k}$$

And $\mathbf{E}[\sum_{j=1}^{m} Z_j] = 4(n^{\epsilon} + k)$. By Chernoff bound, $\Pr[\sum_j Z_j < 2(n^{\epsilon} + k)] \leq e^{-\frac{4(n^{\epsilon}+k)}{8}} \leq \frac{1}{100}$. Therefore, the probability that A stores $2(n^{\epsilon} + k)$ samples is at least $99/100$. It's easy to see that A always outputs 1 when the samples are generated from $D_{planted}(x)$ for any x.

The probability that A outputs 1, given that it stored $2(n^{\epsilon} + k)$ samples, when the samples are generate from D_{null} is equal to the probability that there exists a $y \in \{0, 1\}^{n^{\epsilon}+k}$ that could have generated the stored samples. Let $(a'_1, b'_1), ..., (a'_{2(n^{\epsilon}+k)}, b'_{2(n^{\epsilon}+k)})$ be the stored samples. There are at most $2^{n^{\epsilon}+k}$ sequences of $b'_1, ..., b'_{2(n^{\epsilon}+k)}$ generated by some y given

[5] c is a large enough constant

$\{a'_j\}_{j \in [2(n^\epsilon+k)]}$. As, under D_{null}, b is chosen uniformly at random from $\{0,1\}$, the probability that there exists $y \in \{0,1\}^{n^\epsilon+k}$ that could have generated the stored samples is at most $\frac{2^{n^\epsilon+k}}{2^{2(n^\epsilon+k)}} = 2^{-(n^\epsilon+k)} \le 1/100$. Hence, the probability of success is at least 99/100.

5.4 Proof of Theorems 17 and 18

For the full proof, please refer to the full version of the paper [28]. In this section, we give a very brief outline of the proof and state a crucial lemma. Fix a t-resilient k-ary predicate P. Let B be a branching program with bounded width and length, that distinguishes between D_{null} and $D_{planted}(x)$ (x is uniformly distributed over $\{0,1\}^n$) for the predicate P, with non-negligible probability.

We first use hybrid argument to obtain that the branching program B must have a non-trivial probability of distinguishing with a single sample. Towards this, define $H_j(x)$ to be the distribution over m samples where the first j samples are drawn from $D_{planted}(x)$ and the remaining $m - j$ samples are from D_{null}. Using hybrid argument, there is a $j' \in \{1, ..., m\}$ such that B distinguishes between the hybrids $H_{j'-1}(x)$ and $H_{j'}(x)$ with non-negligible probability. Then to contradict, we show that a bounded-width and bounded-length B cannot distinguish between the hybrids $H_{j'-1}(x)$ and $H_{j'}(x)$.

Let v_1 be a vertex of the branching program B after seeing the first $j' - 1$ (generated from a pseudorandom source with a seed x picked uniformly at random from $\{0,1\}^n$). Let $\mathbb{P}_{x|v_1}$ be the probability distribution of the random variable x conditioned on reaching v_1. If the j'th sample is generated using the same pseudorandom source (as in $H_{j'}(x)$), then $\forall a \in [n]^{(k)}$, the bit b is 0 with probability $\sum_{x':P(x'^a)=0} \mathbb{P}_{x|v_1}(x')$ and 1 with probability $1 - \sum_{x':P(x'^a)=0} \mathbb{P}_{x|v_1}(x')$. If the j'th sample is generated using the uniform source (as in $H_{j'-1}(x)$), then $\forall a \in [n]^{(k)}$, the bit b is 0 with probability 1/2 and 1 with probability 1/2. Thus, for any a, B can distinguish with at most $\left|\sum_{x':P(x'^a)=0} \mathbb{P}_{x|v_1}(x') - 1/2\right|$ advantage. We show that when B has bounded width, then with high probability, it reaches a vertex v_1 such that $\mathbb{P}_{x|v_1}$ has high min-entropy. We then use t-resiliency of P to show that when $\mathbb{P}_{x|v_1}$ has high min-entropy, then with high probability over $a \in [n]^{(k)}$, the expression $\left|\frac{1}{2} - \sum_{x':P(x'^a)=0} \mathbb{P}_{x|v_1}(x')\right|$ is small (the following lemma). Hence, with high probability, it's hard for B to distinguish whether the j'th sample was generated from $H_{j'}(x)$ or $H_{j'-1}(x)$. Note that the last $m - j$ samples, generated by a uniform source, can't better this advantage.

Define $T_l = \{\bar{a} \in \{0,1\}^n : \sum_{i=1}^n \bar{a}^i = l\}$ for $l \in \mathbb{N}$. For $\epsilon > 0, k \ge t > 0$, let L be the set of vertices v such that $\mathbb{P}_{x|v}$ has min-entropy of at least $(n - n^\epsilon - t\log(n/t))$, that is, $\forall x' \in \{0,1\}^n$, $\mathbb{P}_{x|v}(x') \le 2^{n^\epsilon - n} \cdot (n/t)^t$.

▶ **Lemma 19.** *For all $0 < \epsilon < 1 - 3\frac{\log 24}{\log n}$, $0 < t \le k < \{\frac{n}{c}, n^{\frac{(1-\epsilon)}{6}}/3\}$, and $v_1 \in L$,*

$$\left|\sum_{x'} \mathbb{P}_{x|v_1}(x') \cdot (-1)^{P(x'^a)}\right| \le c_2 n^{-(\frac{1-\epsilon}{18})t}$$

for all but at most $c_2 n^{-(\frac{1-\epsilon}{18})t}$ fraction of $a \in [n]^{(k)}$ (recall that P is a t-resilient k-ary predicate).

For all $v_1 \in L$, $0 < \epsilon < 1 - 3\frac{\log 24}{\log n}$, $0 < k < \frac{n}{c}$,

$$\left|\sum_{x'} \mathbb{P}_{x|v_1}(x') \cdot (-1)^{Xor(x'^a)}\right| \le 2\left(\frac{n}{k}\right)^{-(\frac{1-\epsilon}{9})k}$$

for all but at most $2(\frac{n}{k})^{-(\frac{1-\epsilon}{9})k}$ fraction of $a \in [n]^{(k)}$.

Here, c and c_2 are large enough constants.

Please refer to the full version [28] for the proof of the above lemma.

References

1 Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Pseudorandom generators in propositional proof complexity. *SIAM J. Comput.*, 34(1):67–88, 2004. doi:10.1137/S0097539701389944.

2 Sarah R. Allen, Ryan O'Donnell, and David Witmer. How to refute a random CSP. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science—FOCS 2015*, pages 689–708. IEEE Computer Soc., Los Alamitos, CA, 2015.

3 Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. Cryptology ePrint Archive, Report 2015/730, 2015. URL: https://eprint.iacr.org/2015/730.

4 Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. *SIAM J. Comput.*, 42(5):2008–2037, 2013.

5 Benny Applebaum. Cryptographic hardness of random local functions - survey. *Computational Complexity*, 25(3):667–722, 2016.

6 Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In *STOC'10—Proceedings of the 2010 ACM International Symposium on Theory of Computing*, pages 171–180. ACM, New York, 2010.

7 Benny Applebaum, Andrej Bogdanov, and Alon Rosen. A dichotomy for local small-bias generators. *J. Cryptology*, 29(3):577–596, 2016.

8 Benny Applebaum and Shachar Lovett. Algebraic attacks against random local functions and their countermeasures. In *STOC*, pages 1087–1100. ACM, 2016.

9 Yonatan Aumann, Yan Zong Ding, and Michael O. Rabin. Everlasting security in the bounded storage model. *IEEE Trans. Information Theory*, 48(6):1668–1680, 2002. doi:10.1109/TIT.2002.1003845.

10 Yonatan Aumann and Michael O. Rabin. Information theoretically secure communication in the limited storage space model. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 65–79, 1999. doi:10.1007/3-540-48405-1_5.

11 Boaz Barak, Zvika Brakerski, Ilan Komargodski, and Pravesh K. Kothari. Limits on low-degree pseudorandom generators (or: Sum-of-squares meets program obfuscation). In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, pages 649–679, 2018. doi:10.1007/978-3-319-78375-8_21.

12 Boaz Barak, Siu On Chan, and Pravesh K. Kothari. Sum of squares lower bounds from pairwise independence [extended abstract]. In *STOC'15—Proceedings of the 2015 ACM Symposium on Theory of Computing*, pages 97–106. ACM, New York, 2015.

13 Boaz Barak and Ankur Moitra. Noisy tensor completion via the sum-of-squares hierarchy. In *COLT*, volume 49 of *JMLR Workshop and Conference Proceedings*, pages 417–445. JMLR.org, 2016.

14 Paul Beame, Shayan Oveis Gharan, and Xin Yang. Time-space tradeoffs for learning finite functions from random evaluations, with applications to polynomials. In *Conference On Learning Theory*, pages 843–856, 2018.

15 Christian Cachin and Ueli M. Maurer. Unconditional security against memory-bounded adversaries. In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, pages 292–306, 1997. doi:10.1007/BFb0052243.

16 Michael S. Crouch, Andrew McGregor, Gregory Valiant, and David P. Woodruff. Stochastic streams: Sample complexity vs. space complexity. In *ESA*, volume 57 of *LIPIcs*, pages 32:1–32:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

17 Mary Cryan and Peter Bro Miltersen. On pseudorandom generators in NC. In *26th International Symposium on Mathematical Foundations of Computer Science, MFCS*, pages 272–284, 2001.

18 Yuval Dagan and Ohad Shamir. Detecting correlations with little memory and communication. In *Conference On Learning Theory*, pages 1145–1198, 2018.

19 Wei Dai, Stefano Tessaro, and Xihu Zhang. Super-linear time-memory trade-offs for symmetric encryption. Cryptology ePrint Archive, Report 2020/663, 2020. URL: `https://eprint.iacr.org/2020/663`.

20 Amit Daniely. Complexity theoretic limitations on learning halfspaces. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 105–117. ACM, 2016. `doi:10.1145/2897518.2897520`.

21 Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. More data speeds up training time in learning halfspaces over sparse vectors. In *NIPS*, pages 145–153, 2013.

22 Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. The complexity of learning halfspaces using generalized linear methods. In *COLT*, volume 35 of *JMLR Workshop and Conference Proceedings*, pages 244–286. JMLR.org, 2014.

23 Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. From average case complexity to improper learning complexity. In *STOC*, pages 441–448. ACM, 2014.

24 Amit Daniely and Shai Shalev-Shwartz. Complexity theoretic limitations on learning dnf's. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, volume 49 of *JMLR Workshop and Conference Proceedings*, pages 815–830. JMLR.org, 2016. URL: `http://proceedings.mlr.press/v49/daniely16.html`.

25 Ilias Diakonikolas, Themis Gouleakis, Daniel M Kane, and Sankeerth Rao. Communication and memory efficient testing of discrete distributions. In *Conference on Learning Theory*, pages 1070–1106, 2019.

26 Stefan Dziembowski and Ueli M. Maurer. On generating the initial key in the bounded-storage model. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 126–137, 2004. `doi:10.1007/978-3-540-24676-3_8`.

27 Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, pages 534–543. ACM, New York, 2002. `doi:10.1145/509907.509985`.

28 Sumegha Garg, Pravesh K Kothari, and Ran Raz. Time-space tradeoffs for distinguishing distributions and applications to security of goldreich's prg. *arXiv preprint*, 2020. `arXiv:2002.07235`.

29 Sumegha Garg, Ran Raz, and Avishay Tal. Extractor-based time-space lower bounds for learning. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 990–1002. ACM, 2018.

30 Sumegha Garg, Ran Raz, and Avishay Tal. Time-space lower bounds for two-pass learning. In *34th Computational Complexity Conference (CCC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

31 Oded Goldreich. Candidate one-way functions based on expander graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(90), 2000.

32 Jiaxin Guan and Mark Zhandary. Simple schemes in the bounded storage model. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 500–524. Springer, 2019.

33 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 406–425, 2011. `doi:10.1007/978-3-642-20465-4_23`.

34 Joseph Jaeger and Stefano Tessaro. Tight time-memory trade-offs for symmetric encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 467–497. Springer, 2019.

35 Gillat Kol, Ran Raz, and Avishay Tal. Time-space hardness of learning sparse parities. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1067–1080. ACM, 2017.

36 Pravesh K. Kothari and Roi Livni. Improper learning by refuting. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 55:1–55:10, 2018. doi:10.4230/LIPIcs.ITCS.2018.55.

37 Pravesh K. Kothari, Ryuhei Mori, Ryan O'Donnell, and David Witmer. Sum of squares lower bounds for refuting any CSP. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 132–145, 2017. doi:10.1145/3055399.3055485.

38 Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, pages 28–57, 2016. doi:10.1007/978-3-662-49890-3_2.

39 Huijia Lin. Indistinguishability obfuscation from sxdh on 5-linear maps and locality-5 prgs. Cryptology ePrint Archive, Report 2016/1096, 2016. URL: https://eprint.iacr.org/2016/1096.

40 Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local prgs. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 630–660, 2017. doi:10.1007/978-3-319-63688-7_21.

41 Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 11–20. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.11.

42 Alex Lombardi and Vinod Vaikuntanathan. Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In *Theory of Cryptography - 15th International Conference, TCC*, volume 10677, pages 119–137. Springer, 2017.

43 Alex Lombardi and Vinod Vaikuntanathan. Minimizing the complexity of Goldreich's pseudorandom generator. *IACR Cryptology ePrint Archive*, page 277, 2017.

44 Ueli M. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *J. Cryptology*, 5(1):53–66, 1992. doi:10.1007/BF00191321.

45 Dana Moshkovitz and Michal Moshkovitz. Mixing implies lower bounds for space bounded learning. In *COLT*, volume 65 of *Proceedings of Machine Learning Research*, pages 1516–1566. PMLR, 2017.

46 Dana Moshkovitz and Michal Moshkovitz. Entropy samplers and strong generic lower bounds for space bounded learning. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

47 Michal Moshkovitz and Naftali Tishby. Mixing complexity and its applications to neural networks. *arXiv preprint*, 2017. arXiv:1703.00729.

48 Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On e-biased generators in NC0. In *FOCS*, pages 136–145. IEEE Computer Society, 2003.

49 Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On ϵ-biased generators in NC^0. *Random Structures Algorithms*, 29(1):56–81, 2006. doi:10.1002/rsa.20112.

50 N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.

51 Ryan O'Donnell and David Witmer. Goldreich's PRG: evidence for near-optimal polynomial stretch. In *IEEE 29th Conference on Computational Complexity—CCC 2014*, pages 1–12. IEEE Computer Soc., Los Alamitos, CA, 2014. `doi:10.1109/CCC.2014.9`.

52 Prasad Raghavendra, Satish Rao, and Tselil Schramm. Strongly refuting random csps below the spectral threshold. In *STOC*, pages 121–131. ACM, 2017.

53 Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 266–275. IEEE, 2016.

54 Ran Raz. A time-space lower bound for a large class of learning problems. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 732–742, 2017.

55 Ohad Shamir. Fundamental limits of online and distributed algorithms for statistical learning and estimation. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 163–171, 2014. URL: `http://papers.nips.cc/book/advances-in-neural-information-processing-systems-27-2014`.

56 Vatsal Sharan, Aaron Sidford, and Gregory Valiant. Memory-sample tradeoffs for linear regression with small error. *arXiv preprint*, 2019. `arXiv:1904.08544`.

57 Jacob Steinhardt, Gregory Valiant, and Stefan Wager. Memory, communication, and statistical queries. In *COLT*, volume 49 of *JMLR Workshop and Conference Proceedings*, pages 1490–1516. JMLR.org, 2016.

58 Stefano Tessaro and Aishwarya Thiruvengadam. Provable time-memory trade-offs: symmetric cryptography against memory-bounded adversaries. In *Theory of Cryptography Conference*, pages 3–32. Springer, 2018.

59 Salil P. Vadhan. On constructing locally computable extractors and cryptosystems in the bounded storage model. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 61–77. Springer, 2003.

60 Salil P. Vadhan. On learning vs. refutation. In *COLT*, volume 65 of *Proceedings of Machine Learning Research*, pages 1835–1848. PMLR, 2017.

61 Gregory Valiant and Paul Valiant. Information theoretically secure databases. *arXiv preprint*, 2016. `arXiv:1605.02646`.