Extractors from Reed-Muller Codes

Amnon Ta-Shma
Department of Computer Science
Tel-Aviv University
Israel 69978.
email: amnon@post.tau.ac.il. *

David Zuckerman
Department of Computer Science
University of Texas
Austin, TX 78712.
email: diz@cs.utexas.edu †

Shmuel Safra
Department of Computer Science
Tel-Aviv University
Israel 69978.
email: safra@post.tau.ac.il

Abstract

Finding explicit extractors is an important derandomization goal that has received a lot of attention in the past decade. Previous research has focused on two approaches, one related to hashing and the other to pseudorandom generators. A third view, regarding extractors as good error correcting codes, was noticed before. Yet, researchers had failed to build extractors directly from a good code without using other tools from pseudorandomness. We succeed in constructing an extractor directly from a Reed-Muller code. To do this, we develop a novel proof technique.

Furthermore, our construction is the first to achieve degree close to linear. In contrast, the best previous constructions brought the log of the degree within a constant of optimal, which gives polynomial degree. This improvement is important for certain applications. For example, it follows that approximating the VC dimension to within a factor of $N^{1-\delta}$ is AM-hard for any positive δ.

1 Introduction

1.1 History and background.

Sipser [Sip88] and Santha [San87] were the first to realize that extractor-like structures can be used to save on randomness. Sipser and Santha showed the existence of such objects, and left open the problem of explicitly constructing them. True extractors were first defined in [NZ96]:

Definition 1.1. *[NZ96] $E : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ is an ϵ-*extractor *for a class of distributions $\mathbb{X}$ over $\{0,1\}^n$, if for every distribution $X \in \mathbb{X}$ the distribution $E(X, U_t)$ is within statistical distance ϵ from U_m.[1] E is explicit if $E(x,y)$ can be computed in time polynomial in the input length $n+t$. E is a (k,ϵ)-*extractor *if E is an extractor for all distributions with min-entropy k.[2]*

Thus, extractors extract the entropy from a defective random source using few additional truly random bits. The goal is to construct extractors for any min-entropy k with t, the number of truly random bits, as small as possible and m, the number of output bits, as large as possible.

Building on earlier work of Zuckerman [Zuc90, Zuc96], Nisan and Zuckerman [NZ96] built an extractor with $t = O(\log^2 n)$ when the entropy of the source k was high, $k = \Omega(n)$. Srinivasan and Zuckerman extended this solution to the case $k = n^{1/2+\epsilon}$ and Ta-Shma [TS96] further extended it to any entropy k. Also, Ta-Shma was the first to

*Some of this work was done while the author was at the University of California at Berkeley, and supported in part by a David and Lucile Packard Fellowship for Science and Engineering and NSF NYI Grant CCR-9457799.

†Some of this work was done while the author was on leave at the University of California at Berkeley. Supported in part by a David and Lucile Packard Fellowship for Science and Engineering, NSF Grant CCR-9912428, NSF NYI Grant CCR-9457799, and an Alfred P. Sloan Research Fellowship.

[1] U_t denotes the uniform distribution on t bits, and $E(X, U_t)$ denotes the distribution obtained by evaluating $E(x,y)$ for x chosen according to X and y according to U_t. Also, see Section 2 for the definition of statistical distance, also known as variation distance.

[2] See Section 2 for the definition of min-entropy.

k	t	m	Ref
$\Omega(n)$	$O(\log n)$	$\Omega(k)$	[Zuc97]
any k	$O(\frac{\log^2 n}{\log k})$	$k^{1-\alpha}$	[Tre99]
any k	$O(\log n)$	$k/\log n$	[RSW00]
$k \geq \sqrt{n}m \log^2 n$	$\log n +$ $O(\log(\log^* m))$	m	This paper
$k \geq n^{1/c}m$	$\log n+$ $O(c^2 \log m)$	m	This paper
$\Omega(n)$	$\log n+$ $O(\log \log n)$	$\Omega(k)$	This paper
any k	$\log n + \Theta(1)$	$k + t$ $-\Theta(1)$	Optimal. [RTS00]

Table 1. Milestones in building explicit extractors. The error ϵ is a constant.

extract all the entropy from the source. Zuckerman [Zuc97] showed a construction with $t = O(\log n)$ working for high entropies $k = \Omega(n)$. All of this work used hashing and k-wise independence in various forms.

Departing from previous techniques, Trevisan [Tre99] showed a connection between pseudorandom generators for small circuits and extractors. Trevisan used the Nisan-Wigderson pseudorandom generator [NW94] to construct a simple and elegant extractor that achieves $t = O(\log n)$ when $k = n^{\Omega(1)}$ and $t = O(\log^2 n)$ for the general case. Trevisan's work was extended in [ISW99, ISW00, RSW00, TSUZ01] to work for every k with only $t = O(\log n)$ truly random bits. These extensions added to the complexity of the extractor.

Thus, in the current state of the art, there are two techniques that are used in various forms and combinations and different degrees of complexity. Even after all that work, all the known constructions use $t = O(\log n)$ while the lower bound is only $t = \log n + O(1)$. This progress is summarized in Table 1.1 for the case of constant error ϵ.

1.2 The signi£cance of the extractor degree.

Besides their straightforward applications to simulating randomized algorithms using weak sources, extractors have had applications to many areas in derandomization that are seemingly unrelated to weak sources. These include constructing expanders that beat the second eigenvalue method [WZ99], superconcentrators and non-blocking networks [WZ99], sorting and selecting in rounds [WZ99], pseudorandom generators for space-bounded computation [NZ96], unapproximability of clique [Zuc96] and certain Σ_2^P minimization problems [Uma99], time versus space complexities [Sip88], leader election [Zuc97, RZ98], another proof that BPP $\subseteq$ PH [GZ97], random sampling using few random bits [Zuc97], and error-correcting codes with strong list decoding properties [TSZ01]. Hastad [Hås96] uses non-explicit dispersers[3] in his result that CLIQUE is unapproximable to within $n^{1-\alpha}$ for any $\alpha > 0$. The use of non-explicit dispersers make the result depend on the assumption that NP $\neq$ ZPP; a derandomized version would assume that NP $\neq$ P.

In many of these applications, extractors are viewed as highly unbalanced strong expanders. In this view an extractor is a bipartite graph $G = (V_1, V_2, E)$ with V_1 being $\{0,1\}^n$, $V_2 = \{0,1\}^m$ and an edge (x, z) exists iff there is some $y \in \{0,1\}^t$ such that $E(x, y) = z$. Thus, the degree of each vertex of V_1 is $T = 2^t$, and the extractor hashes the input $x \in V_1$ to a random neighbor among its T neighbors in V_2.

Often this degree T is of more interest than $t = \log T$. For example, in the samplers of [Zuc97] the degree is the number of samples; in the simulation of BPP using weak sources [Zuc96] the degree is the number of calls to the BPP algorithm; in the extractor codes of [TSZ01] T is the length of the code; and in the unapproximability of CLIQUE, the size of the graph is closely related to T.

As stated before, all previous constructions have degree $T = \text{poly}(n) = \text{poly}(\log |V_1|)$ while the lower bound (that matches non-explicit constructions) is only $T = O(n) = O(\log |V_1|)$. In fact, when the error ϵ allowed is very large and close to 1, dispersers require degree which is smaller than n, namely $O(\frac{n}{\log \frac{1}{1-\epsilon}})$. Getting such a disperser would derandomize Hastad's result.

Our construction breaks the polynomial degree bound and is the £rst to be close to linear. An unapproximability result that required just the extractors we construct is due to Mossel and Umans [MU01]. They showed that if for all $\delta > 0$ there are explicit dispersers with degree $n^{1+\delta}$ for sources with min-entropy n^{δ}, then it is AM-hard to approximate the VC dimension to within a factor of $N^{1-\gamma}$ for all positive γ. Our degree is even smaller, and extractors are stronger than dispersers, so the unconditional unapproximability of VC dimension follows.

1.3 Our construction.

Our construction uses error-correcting codes. Codes are known to be related to extractors. In fact, Ta-Shma and Zuckerman [TSZ01] showed that extractors are equivalent to codes over large alphabets having good list decoding properties. Explicit extractors correspond to codes with explicit encoding. However, they used extractors to give good list decodable codes, but not the reverse.

[3] A disperser is a one-sided version of extractor.

Even earlier, Trevisan used error-correcting codes in his extractor construction. However, this construction seems to draw its power from pseudorandom generators rather than from coding theory. Indeed, Trevisan himself emphasized the use of pseudorandom generators, since he obtained relatively strong extractors without using codes at all.

Trevisan's extractor can be viewed as £rst encoding the weak source input $x \in \{0,1\}^n$ with any good error correcting code (good here means minimum distance close to half) and then using the truly random bits y to select bits from the encoded string using designs. A natural question is whether one can use a speci£c good code to allow y to be used in a more ef£cient way. Our extractor construction is the £rst to do this.

Our good code is a Reed-Muller code. Speci£cally, we view the input x from the weak source as de£ning a degree h multivariate polynomial $\hat{x} : \mathbb{F}^D \rightarrow \mathbb{F}$ over some large £eld $\mathbb{F}$ (h is about $n^{1/D}$). We will need to convert £eld elements to bits, for which we use a good binary code. We use the truly random bits to choose an element $a \in \mathbb{F}^D$, as well as a random position j of the binary code. The ith output bit of the extractor is the jth bit of $\hat{x}(a + (i, 0, \ldots, 0))$. Thus, the construction can be viewed as using a Reed-Muller concatenated with a good binary code for encoding, and selecting m consecutive points for the output.

For simplicity we will focus on the bivariate case, though the multivariate case works as well, and gives different parameters. Notice the simple way in which we use the random string y to select the output bits. Indeed, this gives an extractor using only $t = \log n + O(\log \frac{m}{\epsilon})$ truly random bits. Formally, we prove:

Theorem 1. *For every $m = m(n)$, $k = k(n)$ and $\epsilon = \epsilon(n) \leq 1/2$ such that $3m\sqrt{n \log(n/\epsilon)} \leq k \leq n$, there is an explicit family of (k, ϵ) extractors $E_n : \{0,1\}^n \times \{0,1\}^t \rightarrow \{0,1\}^m$ with $t = \log n + O(\log \frac{m}{\epsilon})$.*

To further improve our result, we develop a technique to dramatically reduce the factor of m penalty that Yao's lemma usually induces in the error. In particular, we notice that our construction (and Trevisan's, for example) can be viewed as an ef£cient reduction from the problem of constructing extractors for general sources, to the problem of constructing extractors for almost semi-random sources (de£ned in Section 3.1). We then show an ef£cient construction for almost semi-random sources. We use this to give a construction using only $t = \log n + O(\log(\log^* m))$ truly random bits. Formally,

Theorem 2. *For every $m = m(n)$, $k = k(n)$ and $\epsilon = \epsilon(n)$ such that $3m\sqrt{n \log(n/\epsilon)} \leq k \leq n$ and $m \geq \Omega(\log^2(\frac{1}{\epsilon}))$, there is an explicit family of (k, ϵ) extractors $E_n : \{0,1\}^n \times \{0,1\}^t \rightarrow \{0,1\}^m$ with $t = \log n + \mathrm{polylog}(\epsilon^{-1}) + O(\log(\log^* m))$* [4].

Notice that while we dramatically improve the random bit complexity, both constructions work only for entropies $k = n^{1/2+\gamma}$ and the number of output bits is only k^δ for some $\delta < \gamma$. We can make the construction work for smaller entropies by using multivariate polynomials instead of bivariate polynomials, but then we pay in the number of the output bits and the error. Namely, we prove:

Theorem 3. *For every n, m and D, there is an explicit family of (k, ϵ) extractors $E_n : \{0,1\}^n \times \{0,1\}^t \rightarrow \{0,1\}^m$ with $k = \Omega(m^{D-1} n^{1/D} \log n)$, $t \leq \log n + O(D^2 \log m)$, and $\epsilon = 1 - \frac{1}{8D}$.*

We defer the proof to the full version of the paper.

We can also extract more output bits in the case where the entropy is $k = \Omega(n)$. Formally,

Theorem 4. *For any constants $\delta, \epsilon > 0$, there is a constant γ such that there is an explicit family of $(\delta n, \epsilon)$ extractors $E : \{0,1\}^n \times \{0,1\}^t \rightarrow \{0,1\}^m$, where $t = \log n + O(\log \log n)$ and $m = \gamma n$.*

Beyond the signi£cance of the better bounds on the degree of Theorems 2,3 and 4, Theorem 1 is the £rst (and only) purely algebraic extractor construction, and the only one relying solely on error-correcting codes. Moreover, the simplicity and good constants may make it practical, which is probably not the case for previous constructions.

1.4 Our proof technique.

At the highest level our proof plan resembles Trevisan's. That is, we assume that the distribution $E(X, U_t)$ is not close to uniform. Using Yao's lemma this gives a next bit predictor, which we use to give a small description of the elements of the support of X. In our case, the next bit predictor on average can learn the value $\hat{x}(a_1 + i, a_2)$ from the values $\hat{x}(a_1 + 1, a_2), \ldots, \hat{x}(a_1 + i - 1, a_2)$, where $\hat{x} : \mathbb{F}^2 \rightarrow \mathbb{F}$ is a bivariate polynomial.

The similarity to Trevisan's proof ends there. We now play a mental game. We pick a random line L and we assume someone is giving us the correct values of $\hat{x}$ on $i - 1$ consecutive parallel left-shifts of L, i.e., $L - (1, 0)$ through $L - (i - 1, 0)$. Each point on L is preceded by $i - 1$ points for which we already know (by assumption) the correct value of $\hat{x}$. Hence we can use the predictor to predict that point, with some moderately good success probability. Overall, the predictor is correct for many points on L. We now use the fact that $\hat{x}$ restricted to L is a low-degree polynomial to actually £nd the value of $\hat{x}$ on that line (using list-decoding for

[4] The $\mathrm{polylog}(\epsilon^{-1})$ term can be improved to $O(\log \epsilon^{-1})$ but we postpone it to the full version of the paper.

Reed-Solomon codes). We then learn $L+(1,0)$, $L+(2,0)$, etc., until we learn enough lines to reconstruct $\hat{x}$ itself.

Playing that mental game we can prove that there is a set of about mh queries (and recall that h is about $\sqrt{n}$ and m is the number of output bits) such that almost every $x \in X$ can be reconstructed given the answers to these queries. Note that h^2 queries can always reconstruct X; our gain is that we can reconstruct X using only mh instead of h^2 queries. This shows that X has size at most $|\mathbb{F}|^{mh}$, or equivalently has entropy at most $mh \log q$. We conclude that if X is larger than that the distribution $E(X,U)$ is close to uniform.

The technique was inspired by work done on list decoding of Reed-Muller codes, and its application to hardness ampli£cation in [STV99]. Aside from applying an error-correcting technique to a different information theoretic setting, our proof techniuqe has additional ideas. For example, we obtain our savings by learning a line using previously learned lines, and this whole notion of recycling queries makes sense only in our setting and does not appear in previous constructions.

We believe this is a clean and elegant way of constructing extractors. Indeed, there has already been another construction based on ours [SU01].

2 Preliminaries

Throughout, $\mathbb{F} = \mathbb{F}_q$ denotes a £eld of size q. As usual, $[n]$ denotes the set $\{1,2,\dots,n\}$. If S is a set and a is an element, then $S+a$ denotes the set $\{s+a : s \in S\}$. All logarithms are to the base 2. We will assume, when needed, that various quantities are integers. It is not hard to check that this has only a negligible effect on our analysis.

The *statistical distance* (or *variation distance*) between two distributions D_1 and D_2 on the same space S is $\max_{T\subseteq S} |D_1(T) - D_2(T)| = \frac{1}{2}\sum_{s\in S} |D_1(s) - D_2(s)|$. The *min-entropy* of a distribution D on a probability space S is $\min_{s\in S}\{-\log_2 D(s)\}$.

An element $x \in \Lambda^1 \times \cdots \times \Lambda_m$, for some domains Λ_i, is an m–tuple, $x = x_1 \circ \ldots \circ x_m$. Similarly, a distribution X over Λ^m, de£nes m correlated distributions $X = X_1 \circ \ldots \circ X_m$. For $1 \le i \le j \le m$ let $X_{[i,j]}$ denote the random variable $X_i \circ \ldots \circ X_j$ and similarly for x. Thus, $\Pr(X_{[i,j]} = x_{[i,j]})$ is a shortcut for $\Pr(X_i = x_i \wedge \ldots \wedge X_j = x_j)$.

POLYNOMIALS. We will use the following lemma due to Sudan.

Lemma 2.1. *[Sud97] Given a sequence of m distinct pairs pairs $\{x_i, y_i\} \in \mathbb{F}^2$, there are less than $2m/a$ degree h polynomials p such that $p(x_i) = y_i$ for at least a values of $i \in [m]$, provided that $a \ge \sqrt{2hm}$.*

Corollary 2.2. *For each element $u \in \mathbb{F}$ assign a set S_u of size at most A. Then there are less than $2A/\delta$ degree h polynomials p such that $p(u) \in S_u$ for at least a δ fraction of points, provided that $\delta \ge \sqrt{2hA/|\mathbb{F}|}$.*

BINARY CODES. An $[n,k]$ code is a linear binary code of length n and dimension k. When we refer to relative measures, we mean the ordinary measures divided by the length. We will need binary codes with good combinatorial list decoding properties.

De£nition 2.3. *A binary code has combinatorial list decoding property α if every Hamming ball of relative radius $\frac{1}{2} - \alpha$ has $O(1/\alpha^2)$ codewords.*

We will use codes from the following code construction due to [GHSZ00].

Fact 2.4. *There is a polynomial-time (in fact, Logspace) constructible $[n,k]$ code with combinatorial list decoding property α, where $n = O(k/\alpha^4)$.*

Simpler and more ef£cient constructions can be achieved with somewhat worse parameters, e.g. [NN93, AGHP92].

Our construction uses an (h,D) Reed-Muller code over $\mathbb{F}_q$. In such a code the message speci£es a polynomial f in D variables over $\mathbb{F}_q$ of total degree at most h, and the output is all the values of f over $\mathbb{F}_q^D$. Every polynomial in D variables of total degree $\le h$ can be represented by the coef£cients of the different monomials $x_1^{i_1} \cdot \ldots \cdot x_D^{i_D}$ with $i_1 + \ldots i_D \le h$, and there are exactly $\binom{h+D}{D}$ such monomials. It follows that such a code has length q^D and dimension $\binom{h+D}{D}$.

3 Top-down overview

3.1 Almost Semi-Random Sources

Semi random sources were de£ned and studied in many early papers, most notably [SV86, CG88, NZ96, SZ99]. We extend this de£nition to a β–almost semi-random source:

De£nition 3.1. *A distribution $Z = Z_1 \circ Z_2 \circ \ldots \circ Z_m$ is β–almost α semi-random if for every $i = 1,\dots,m$, Z_i is distributed over $\{0,1\}$ and*

$$\Pr_{z\in Z}[\Pr_{z_i}(Z_i = z_i | Z_{[1,i-1]} = z_{[1,i-1]}) > \frac{1}{2} + \alpha] \quad \le \quad \beta$$

If $\beta = 0$ we say Z is α semi-random.

Two simple facts are recorded in the following lemma.

Lemma 3.2. *A β–almost α semi-random source $Z = Z_1 \circ \ldots \circ Z_m$ is $m\beta$ close to an α semi random source. An α semi-random source $Z = Z_1 \circ \ldots \circ Z_m$ is $m\alpha$ close to the uniform distribution.*

Proof. For the £rst part, we induct from $i = m$ down to $i = 1$. For each such i there is at most a β fraction of bad pre£xes. For each such bad pre£x, we can redistribute the weight of strings with that pre£x uniformly on all extension of the pre£x. The redistribution for a particular i makes at most a β difference to the distribution, and the £nal distribution is a true semi-random source.

For the second part, notice that for every pre£x the probability of the next bit being 0 or 1 is at most $\frac{1}{2} + \alpha$. We can again induct from $i = m$ to $i = 1$ and redistribute the weight so that it is perfectly uniform. The resulting difference is again at most $m\alpha$. □

An extractor working on a general distribution over n input bits requires at least $t \geq \log n - O(1)$ truly random bits, even when the allowed error ϵ is a constant. In contrast, we show that extractors for semi-random sources require only $t = O(1)$ truly random bits and explicitly construct such an extractor. We defer the proof to Section 5.

Theorem 5. *For every $\alpha, \beta > 0$ de£ne $p = 1 - \log(1+2\alpha)$. There is an ϵ-extractor $F : \{0,1\}^m \times \{0,1\}^r \to \{0,1\}^{m'}$ for β-almost α semi-random sources with $\epsilon = O(\beta \log^* m)$, $r = \mathrm{poly}(\log \frac{1}{p\epsilon})$ and $m' = \frac{pm}{2} - O((\log^* m)^2 \log \epsilon^{-1})$.*

3.2 The main construction

Let $\mathbb{F} = \mathbb{F}_q$ be a £eld of size $q \gg \sqrt{n}$. De£ne $h = \left\lceil \sqrt{\frac{2n}{\log q}} \right\rceil$. It is easiest to think of q as prime. The reduction begins by viewing the n-bit input string $x \in \{0,1\}^n$ as a function $f : \mathbb{F}^2 \to \mathbb{F}$ of total degree at most $h - 1$. This is possible since every degree $h-1$ bivariate polynomial f can be speci£ed using $\binom{h+1}{2}$ coef£cients from $\mathbb{F}_q$ and $\binom{h+1}{2} \log q \geq \frac{h^2}{2} \log q \geq n$.

The function $\mathbf{Z}$

Input : $x \in \{0,1\}^n$.

Setting : $\mathbb{F}$ is a £eld of size q, $h = \left\lceil \sqrt{\frac{2n}{\log q}} \right\rceil$. We associate with $x \in \{0,1\}^n$ a function $\hat{x} : \mathbb{F}^2 \to \mathbb{F}$ of total degree at most $h - 1$.

Binary code : **BC** is a linear binary code of dimension $\ell = \log q$, combinatorial list decoding property α (see De£nition 2.3), α will be determined later, and length $\bar{\ell}$.

Random coins : $a = (a_1, a_2) \in \mathbb{F}^2$, $j \in [\bar{\ell}]$.

Output : $\mathbf{Z}(x; a, j)_i = \mathbf{BC}(\hat{x}(a_1 + i, a_2))_j$ for $i = 1, \ldots, m$

Our main technical contribution is:

Theorem 6. *For every n, m and $\alpha, \beta > 0$:*

- *Set q the smallest prime such that $q^2 \log q \geq (\frac{2^9\sqrt{n}}{\alpha^2\beta^2})^2$.*
- *Set $h = \left\lceil \sqrt{\frac{2n}{\log q}} \right\rceil$.*

Let us denote $\mathbf{Z} = \mathbf{Z}(X, U_t)$ where $\mathbf{Z} = \mathbf{Z}_1 \circ \mathbf{Z}_2 \circ \ldots \circ \mathbf{Z}_m$. Then:

- *If $\mathbf{Z}$ is not β-almost α semi-random, $|X| < q^{mh}$.*
- *$t \leq \log n + O(\log(\frac{1}{\alpha\beta}))$, and,*
- *$\mathbf{Z}$ can be computed in $O(\log q)$ space and $\mathrm{poly}(q) = \mathrm{poly}(n, \frac{1}{\alpha\beta})$ time.*

To verify parameters note that $\mathbf{Z}$ uses $t = \log(q^2) + \log \bar{\ell}$ random bits. Using Fact 2.4, $\bar{\ell} = O((\log q)/\alpha^4)$, so $t = \log(q^2 \log q) + O(\log \alpha^{-1}) = \log n + O(\log \frac{1}{\alpha\beta})$.

The running time of $\mathbf{Z}$ is dominated by the complexity of evaluating $\hat{x}(a)$, which can be done with $O(\log q)$ space and $\mathrm{poly}(q)$ time. We will deal later with the challenging part of proving the reduction correctness.

We are now ready to prove Theorem 1:

Proof. (Of Theorem 1) Suppose $m = m(n)$, $k = k(n)$ and $\epsilon = \epsilon(n)$ are such that $3m\sqrt{n \log(\frac{n}{\epsilon})} \leq k \leq n$. Let us set $\alpha = \beta = \frac{\epsilon}{2m}$ and q and h as in Theorem 6. We claim $\mathbf{Z}(X; U)$ is the desired (k, ϵ) extractor.

To see that, let $X \subseteq \{0,1\}^n$ be an arbitrary set of cardinality at least $K = 2^k$. As $q^2 \leq o(n(\frac{m}{\epsilon})^8) \leq o(\frac{n}{\epsilon})^9$, one can check that for large n, $K = 2^k \geq q^{mh}$. By Theorem 6, $\mathbf{Z} = \mathbf{Z}(X; U)$ is a β–almost α semi-random source. It follows by Lemma 3.2 that $\mathbf{Z}(X; U)$ is $m(\alpha + \beta) = \epsilon$ close to uniform. □

3.3 Further reducing the number of truly random bits

We now want to save the $O(\log m)$ extra term we have in t. To do that we work with a constant α, say $\alpha = .1$. Then the distribution $\mathbf{Z} = \mathbf{Z}(X; U)$ is not close to uniform. However, $\mathbf{Z}$ is β–almost α semi-random, with constant entropy in each of the bits. It therefore suf£ces to give an extremely ef£cient extractor for almost semi-random sources. We now prove Theorem 2.

Proof. (Of Theorem 2) Suppose $m = m(n)$, $k = k(n)$ and $\epsilon = \epsilon(n)$ are such that $3m\sqrt{n \log(n/\epsilon)} \leq k \leq n$ and $m \geq 2\log^2 \frac{1}{\epsilon}$. Let us set $\alpha = .1$, $\beta = \Theta(\frac{\epsilon}{\log^* m})$ and $q = \Theta(\sqrt{\frac{n}{\log n}} \frac{1}{\beta^2})$. Let $\mathbf{F}(X; U)$ be the extractor for β–almost semi-random sources, of Theorem 5. Our extractor is $E(x; y_1, y_2) = \mathbf{F}(\mathbf{Z}(x; y_1); y_2)$. That is, we £rst apply $\mathbf{Z}$

on the input x together with the truly random string y_1, and then we apply the extractor $\mathbf{F}$ for β–almost semi-random sources together with a new truly random string y_2.

Correctness : Let $X \subseteq \{0,1\}^n$ be an arbitrary set of cardinality at least $K = 2^k$. Working out the parameters we see that $|X| \geq 2^k \geq q^{mh}$. By Theorem 6, $\mathbf{Z} = \mathbf{Z}(X;U)$ is a β–almost .1 semi-random source. It then follows by Theorem 5 that $E(X;U)$ is $O(\epsilon)$ close to uniform.

Parameters : By Theorem 6, the length of y_1 is $\log n + 4\log(\frac{1}{\beta}) + O(1) = \log n + \log(\log^* m)) + \log\frac{1}{\epsilon} + O(1)$. By theorem 5 the length of y_2 is $\mathrm{polylog}(\epsilon^{-1})$. Thus the number of truly random bits used is as required. The output length of $\mathbf{Z}$ is m, and thus the output length of F is at least $\frac{pm}{2} - O((\log^* m)^2 \log\frac{1}{\epsilon}) \geq \frac{pm}{4}$ provided that $m \geq \Omega(\log^2 \frac{1}{\epsilon})$. To get error ϵ and m output bits, just apply the above construction with $m' = 4m$ and $\epsilon' = \Omega(\epsilon)$.

□

3.4 Increasing the output length

We can apply techniques from [RSW00, NZ96] to increase the length of the output and prove Theorem 4:

Proof. (of Theorem 4, Sketch.) Reingold, et.al. [RSW00] showed how to add $O(\log\log n)$ truly random bits to the input X and extract a block B of length $\frac{\delta}{2}n$, such that B is close to a $2\gamma n$ source and X is close to a source with min-entropy $\frac{\delta}{4}n$, even conditioned on B. We can then apply techniques implicit in [NZ96] and explicit in [SZ99]. We use the extractor of Theorem 2 to add $\log n + O(\log^* n)$ truly random bits to X and extract $\mathrm{polylog}(n)$ almost-random bits that are close to uniform even conditioned on B. We now use these $\mathrm{polylog}(n)$ bits and the original extractor of [NZ96] (though the more complex [Zuc97] is better) to extract γn almost-random bits from B. □

4 The main proof

To prove Theorem 6, we show that if $\mathbf{Z}$ is not as required then there exists a large subset X'' of X such that the answers to a small number of queries distinguishes elements of X''. This shows that X'' is small, and therefore X itself is small. We begin with some de£nitions.

De£nition 4.1. *A predictor P for bivariate polynomials is a probabilistic function which, on input $a \in F^2$, makes queries $Q(a)$ about a bivariate polynomial f. The set $Q(a)$ may be chosen at random and may or may not depend on a. On receiving the answers to the queries from an oracle, P computes a subset $P(a) \subseteq F$.*

The queries are usually point queries of the form, "what is $f(v)$?" for some point $v \in F^2$. Some queries are line queries of the form "which polynomial among $p_1, \ldots, p_B$ is equal to f restricted to line L, denoted $f|_L$?" We will be interested in the number of possibilities a query distinguishes. For example, this is q for the point queries and B for the line queries. The number of possibilities a set of queries distinguishes is the product of the possibilities for each query in the set.

P has A possible answers if for every $a \in F^2$, every possible set of queries and every set of answers, the size of $P(a)$ is at most A. P predicts $f : F^2 \to F$ with success p if, when the oracle answers the queries according to f,

$$\Pr_{\substack{a \in F^2 \\ \text{coins of } P}} [f(a) \in P(a)] \geq p$$

P predicts $S \subseteq \{0,1\}^n$ with success p if, for every $x \in S$, P predicts $\hat{x}$ with success p. P has preprocessed queries if $Q(a)$ does not depend on a, and P is deterministic if it does not use random coins (neither to choose Q nor to compute its answer).

Notation 4.2. *Suppose P is a predictor for bivariate polynomials and $f : \mathbb{F}^2 \to \mathbb{F}$. $P^f_Q(a)$ denotes the value when the queries are $Q(a)$ and the oracle answers according to f. Whenever the set of queries Q is clear from the context we write $P^f(a)$ to denote this value. If P is deterministic, the set of queries Q is completely determined by a, and then we always denote this value by $P^f(a)$.*

Theorem 6 follows from the following proposition:

Proposition 1. *If $\mathbf{Z} = \mathbf{Z}(X;U)$ is not β–almost α semi-random, then there exists a deterministic predictor for a set X'', of size at least $|X''| \geq \frac{\beta}{4}|X|$, with 1 possible answer and success 1. There are at most $(m-1)h$ point queries and h line queries, all preprocessed, and the line queries distinguish at most $O(1/\beta\alpha^2) = o(\beta q)$ possibilities. Hence $|X| \leq o(q^{mh})$.*

Notice that while in general we need about $\frac{h^2}{2}$ values to determine an arbitrary degree $h-1$ bivariate polynomial, here only about mh queries suf£ce. This immediately implies X is small.

4.1 Evaluating a Point

We now £x a set X such that $\mathbf{Z} = \mathbf{Z}(X;U)$ is not β–almost α semi-random. We begin with a certain "segment predictor."

De£nition 4.3. *P is a segment predictor if the queries are point queries of the form $Q(a_1, a_2) = \{(a_1 - s, a_2), \ldots, (a_1 - 1, a_2)\}$.*

Our £rst step towards proving Proposition 1 will be to prove the following.

Lemma 4.4. *There exists a segment predictor* **EP** *for $X' \subset X$ of cardinality $|X'| \geq \frac{\beta}{2}|X|$ making at most $m-1$ point queries, $A = O(1/\alpha^2)$ possible answers and $p = \frac{\beta}{4}$ success.*

Proof. Since $\mathbf{Z}$ is not β–almost α semi-random, then there is an $i_0 \in \{1, \ldots, m\}$ such that

$$\Pr_{z \in \mathbf{Z}}[\Pr(\mathbf{Z}_{i_0} = z_{i_0} | \mathbf{Z}_{[1,i_0-1]} = z_{[1,i_0-1]}) > \frac{1}{2} + \alpha] \quad \geq \quad \beta$$

We can now de£ne $T : \{0,1\}^{i_0-1} \to \{0,1\}$ by letting $T(z_1, \ldots, z_{i_0-1})$ be the most frequent value of $(\mathbf{Z}_i \mid \mathbf{Z}_{[1,i_0-1]} = z_{[1,i_0-1]})$ breaking ties arbitrarily. We then have $\Pr_{z \in \mathbf{Z}}[\Pr(T(z_{[1,i_0-1]}) = z_{i_0}) > \frac{1}{2} + \alpha] \geq \beta$. An averaging argument shows that there exists a subset $X' \subset X$ of cardinality at least $\frac{\beta}{2}|X|$ such that for every $x \in X'$,

$$\Pr_{\substack{u \in U \\ z \in \mathbf{Z}(x;u)}}[\Pr(T(z_{[1,i_0-1]}) = z_{i_0}) > \frac{1}{2} + \alpha] \quad \geq \quad \frac{\beta}{2}$$

Another averaging argument yields, for every $x \in X'$

$$\Pr_a[\Pr_j[\Pr(T(z_{[1,i_0-1]}) = z_{i_0}) > \frac{1}{2} + \alpha] \geq \frac{\beta}{4}] \quad \geq \quad \frac{\beta}{4} \tag{1}$$

We can now de£ne our segment predictor.

<u>**EP** *: Evaluate Point*</u>

Input : $a = (a_1, a_2) \in \mathbb{F}^2$

Queries : The query points are $Q(a_1, a_2) = \{(a_1 - (i_0 - 1), a_2), \ldots, (a_1 - 1, a_2)\}$. The answer to the query $(a_1 - i, a_2)$ is b_i.

Algorithm : For every $j \in [\bar{\ell}]$ compute

$$g_j(a) \quad = T(\mathbf{BC}(b_{i_0-1})_j, \ldots, \mathbf{BC}(b_1)_j)$$

and set $g(w) = g_1(w) \ldots g_{\bar{\ell}}(w)$.

Output : $\mathbf{EP}(w)$ is the set of all codewords of **BC** that have at least $\frac{1}{2} + \alpha$ relative agreement with $g(w)$.

Note that **EP** is deterministic and that the queries depend on the input a. We claim:

Claim 4.5. *For every $x \in X'$ we have* $\Pr_a(\hat{x}(a) \in \mathbf{EP}^{\hat{x}}(a)) \geq \frac{\beta}{4}$.

Proof. When the answers $b_1, \ldots, b_{i_0-1}$ re¤ect the values of $\hat{x}$ we have $b_i = \hat{x}(a_1 - i, a_2)$ and:

$$g_j(a) \quad = T(\mathbf{BC}(\hat{x}(a_1 - i_0 + 1, a_2))_j, \ldots, \mathbf{BC}(\hat{x}(a_1 - 1, a_2))_j)$$

By Equation (1) for every $x \in X'$, for at least $\frac{\beta}{4}$ of the points $a \in \mathbb{F}^2$, $\Pr_{j \in [\bar{\ell}]}[g_j(a) = \mathbf{BC}(\hat{x}(a))_j] \geq \frac{1}{2} + \alpha$ and whenever this happens we have that $\hat{x}(a) \in \mathbf{EP}^{\hat{x}}(a)$. ☐

Finally, since **BC** has combinatorial list decoding property α for any a, $|\mathbf{EP}^{\hat{x}}(a)| \leq O(\frac{1}{\alpha^2})$. ☐

4.2 Evaluating a Line

We use the procedure **EP** to build a procedure **EL** (for "Evaluate-Line") that given a line L makes some speci£c queries and outputs a single polynomial over $\mathbb{F}$.

<u>$\mathbf{EL}^p(L)$ *: Evaluate Line*</u>

Input : A line $L : \mathbb{F} \to \mathbb{F}^2$.

Algorithm :

- For every $i = 1, \ldots, q$:
  - Evaluate $\mathbf{EP}(L(i))$ by making the necessary point queries.
- Form the set $S = \{(i, w) \mid i \in [1, q] \,,\, w \in \mathbf{EP}(L(i))\}$.
- Compute the list G of all univariate polynomials $g : \mathbb{F} \to \mathbb{F}$ of degree at most $h - 1$ with agreement at least $\frac{\beta}{8}q$ with S.

Output : Query which one of the polynomials in G is equal to $f|_L$, and output this polynomial.

We now show that **EL** does well on random lines.

Lemma 4.6. *For every $x \in X'$:*

$$\Pr_{L, j_1, \ldots, j_\ell}[\mathbf{EL}^{\hat{x}}(L) \neq \hat{x}(L)] \quad \leq \quad \eta = \frac{16}{\beta q}$$

The lemma follows from the following claim, which shows that G almost always contains the right polynomial.

Claim 4.7. *For every $x \in X'$,* $\Pr_L \ (\hat{x}(L) \notin G) \ \leq \ \frac{16}{\beta q}$

Proof. Call $v \in \mathbb{F}^2$ *nice* for $p : \mathbb{F}^2 \to \mathbb{F}$ if $p(v) \in \mathbf{EP}^p(v)$. We know that:

- For every $x \in X'$, $\Pr_{v \in \mathbb{F}^2}[v \text{ is nice for } \hat{x}] \geq \frac{\beta}{4}$ and
- For every $p : \mathbb{F}^2 \to \mathbb{F}$ and v, $|\mathbf{EP}^p(v)| \leq A = O(\frac{1}{\alpha^2})$.

Fix any $x \in X'$. Let Y_i be the random variable indicating whether $L(i)$ is nice, and $Y = \sum_{i=1}^{q} Y_i$. We have $E(Y) \geq \frac{\beta}{4}q$, i.e., for every $x \in X'$ we expect to see many nice points on a random line L. We say L is bad for x if $Y \leq \frac{\beta}{8}q$. As L is a random line the points on L are pairwise independent and it follows that

$$\Pr[Y \leq \frac{E(Y)}{2}] \leq \frac{4\sigma^2(Y)}{(E(Y))^2} \leq \frac{4}{E(Y)} = \frac{16}{\beta q}$$

If the line L is bad for x (which as we saw happens with probability $O(\frac{1}{\beta q})$) we lose. Otherwise, $Y > \frac{E(Y)}{2}$ and the line L contains at least $\frac{\beta}{8}q$ nice points $v = L(i)$. Therefore, $\hat{x}(L) \in G$. □

We now claim that the list decoding process does not return many possible solutions. To see that notice that since $|S| \leq qA \leq O(\frac{q}{\alpha^2})$ and $q \geq \frac{2^9 h}{\alpha^2 \beta^2}$ we have $\frac{\beta q}{8} \geq \sqrt{2h \cdot |S|}$ and therefore by Lemma 2.1, $|G| \leq \frac{2qA}{\beta q/8} = O(\frac{1}{\beta\alpha^2})$.

We now take a closer look at the point queries done in $\mathbf{EL}(L)$.

Lemma 4.8. *If L is not parallel to the x-axis then $\mathbf{EL}(L)$ makes point queries for h points on each of the $i_0 - 1$ lines $L - (j, 0)$, $j \in [i_0 - 1]$.*

Proof. As written, $\mathbf{EL}(L)$ queries all points on these lines. The lemma follows since the values of $\hat{x}$ on a line can be determined by querying h points on that line. □

Thus, $\mathbf{EL}^{\hat{x}}$ makes $(i_0 - 1)h$ point queries.

4.3 Proof of Proposition 1

Proof. We now give a procedure $\mathbf{EA}$ (for "Evaluate-All") that makes few queries and outputs, with good probability, the unique polynomial $\hat{x} \in \hat{X}'$ that agrees with the queries.

<u>**EA** *: Evaluate All*</u>

Input : none.

Algorithm : Pick a random line L, and query the points in $LS(L)$.

For $j = 0$ to $h - i_0$

- Evaluate $\mathbf{EL}(L+(j,0))$. Note that all of the point queries needed to evaluate this have been made or deduced previously. Only the line queries need to be made.

Output the unique degree $h - 1$ polynomial consistent with the answers on the $h \times h$ block.

Now $\mathbf{EA}$ makes $(i_0-1)h$ point queries and $h-i_0+1$ line queries. We de£ne the "shape" SH for a line L to be the set of point queries made by $\mathbf{EA}$ when $\mathbf{EA}$ picks the random line L. We say SH is a shape, if it is the shape for some line L. We say that a shape SH is good for $f : \mathbb{F}^2 \to \mathbb{F}$ if $\mathbf{EA}^f_{SH}$, the output of $\mathbf{EA}^f$ when picking the shape SH, is f, and bad otherwise.

Claim 4.9. *For every $x \in X'$, $\Pr_{SH}[SH$ is bad for $\hat{x}] \leq h\eta$.*

Proof. For each of the h lines we learn, the probability we fail (given the right answers to the shape we use) is at most η. By the union bound (regardless of correlations) the claim follows. □

Now, $h\eta \leq \frac{16}{\beta}\frac{h}{q} \leq \frac{16}{\beta}\frac{\beta^2}{2^9} \leq \frac{1}{2}$, thus, for every $x \in X'$, $\Pr_{SH}[SH$ is good for $\hat{x}] \geq \frac{1}{2}$. Hence, there is at least one £xed choice of a shape SH and a subset $X'' \subseteq X'$ of cardinality at least $|X''| \geq \frac{1}{2}|X'|$ such that SH is good for every $x \in X''$.

Note that the number of point queries is at most $(i_0 - 1)h \leq (m-1)h$, and the number of line queries is at most $h - i_0 + 1 \leq h$. Also, the line queries distinguish at most $O(1/(\beta\alpha^2) \leq o(\beta q)$ possibilities, where the inequality uses the choice of q. Thus the predictor distinguishes at most $q^{(m-1)h}(o(\beta q))^h \leq o(\beta q^{mh})$ possibilities. Proposition 1 and hence Theorem 6 follow. □

5 Extractors for almost semi-random sources

Before we start we need two tools. One is the ([RRV99],Theorem 4) family of extractors: $E_n : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ which is a (k,ϵ) extractor, with $t \leq c_0 \log^3 n \log \epsilon^{-1}$ and $m = k + t - 2\log\epsilon^{-1} - O(1)$, for some constant c_0.

We also need a generalization of semi-random sources to block-wise sources in which each block W_i may have a different range.

De£nition 5.1. *A distribution $W = W_1 \circ \ldots \circ W_b$ is a β–almost $((n_1,k_1),\ldots,(n_b,k_b))$ block-wise source if for every $i = 1,\ldots,b$, W_i is distributed over $\{0,1\}^{n_i}$ and*

$$\Pr_{w \in W}[\Pr(W_i = w_i \mid W_{[1,i-1]} = w_{[1,i-1]}) > 2^{-k_i}] \quad \leq \quad \beta$$

If $\beta = 0$ we say W is a $((n_1,k_1),\ldots,(n_b,k_b))$ block-wise source.

We use a technique for constructing ef£cient extractors for block-wise sources from [NZ96]. Suppose we are given a $((n_1,k_1),\ldots,(n_b,k_b))$ block-wise source, and b extractors $E_1,\ldots,E_b$ where $E_i : \{0,1\}^{n_i} \times \{0,1\}^{r_i} \to$

$\{0,1\}^{r_{i-1}}$. De£ne $E : \{0,1\}^{n_1+\ldots+n_b} \times \{0,1\}^{r_b} \to \{0,1\}^{r_1}$ by

$$F(z_1,\ldots,z_b;y) \quad \stackrel{\text{def}}{=} \quad E_1(z_1;\ldots E_{b-1}(z_{b-1};E_b(z_b;y))\ldots)$$

Nisan and Zuckerman showed

Lemma 5.2. *[NZ96] If each E_i is a (k_i,ϵ_i) extractor than F is an $\epsilon = \sum_{i=1}^{b} \epsilon_i$ extractor.*

5.1 The semi-random case

Suppose $Z = Z_1 \circ \ldots \circ Z_m$ is α semi-random. We would like to group the m bits into few b blocks, on which we will apply the [NZ96] technique. Indeed, given $\epsilon > 0$ we view Z as a $((n_1,k_1),\ldots,(n_b,k_b))$ block-wise source $Z = W = W_1 \circ \ldots \circ W_b$, where,

- $k_i \geq pn_i$, for $2^{-p} = \frac{1}{2} + \alpha$.
- We de£ne $\epsilon_i = \frac{\epsilon}{2 \cdot 2^{b-i}}$.
- We £x r_b in some way that will be described later.
- For $i < b$, $r_i = r_{i+1} + k_{i+1} - 2\log\frac{1}{\epsilon_i} - O(1)$.
- We let n_i be the largest integer such that
$$r_i \quad \geq \quad c_0 \cdot \log^3 n_i \cdot \log\frac{1}{\epsilon_i} \qquad (2)$$
or if there are not enough remaining bits, just all the remaining bits.

If we now plug in the [RRV99] extractor into the [NZ96] technique, and if indeed (2) holds for every i, then by the [NZ96] result we get an $\sum \epsilon_i \leq \epsilon$ extractor $F : \{0,1\}^m \times \{0,1\}^{r_b} \to \{0,1\}^{r_1}$ with entropy loss at most $O(b\log\epsilon^{-1} + b^2)$.

So now we £x r_b. To ensure that (2) holds for r_b, we let r_b be the £rst integer such that $r_b \geq \max\{(c_0\log\frac{2}{\epsilon})^2, (2\log\frac{1}{\epsilon})^6, (4\log r_b)^6\}$. Working out the parameters we see that $n_b \geq 2^{r_b^{1/6}}$. Also $r_{b-1} \geq k_b \geq 2^{r_b^{1/6}-1}$. Continuing in this manner, we get that $r_i > 2^{r_{i+1}^{1/6}-1}$. This recurrence implies that $b \leq \log^* m + O(1)$ (see e.g., [RZ98]). Summarizing this we get:

Theorem 7. *For every $0 < \alpha, \epsilon < \frac{1}{2}$ set $p = 1 - \log(1+2\alpha)$. There is an ϵ-extractor $F : \{0,1\}^m \times \{0,1\}^r \to \{0,1\}^{m'}$ for α semi-random sources with $r = \mathrm{poly}(\log\frac{1}{p\epsilon})$ and $m' = pm + r - O(\log^* m(\log^* m + \log\frac{1}{\epsilon}))$.*

We remark that using the above technique we can further reduce the number of truly random bits to $O(\log\frac{1}{p\epsilon})$ but we will not use this and we skip the details.

5.2 Reducing a β–almost block-wise source to a block-wise source

We saw that a β–almost semi-random source with m bits is $m\beta$ close to a semi-random source. However, it is important for us to avoid this $m\beta$ penalty. The key observation is that at a price of losing at most half the entropy, if we look at ℓ consecutive bits, then instead of being $\ell\beta$ close to the behavior we expect from a block-wise source, it is 2β close.

Lemma 5.3. *Suppose $Z = Z_1 \circ \ldots \circ Z_m$ is a β–almost α semi-random source. For every $1 \leq a \leq b \leq m$*

$$\Pr_{z \in Z}[\Pr(Z_{[a,b]} = z_{[a,b]} | Z_{[1,a-1]} = z_{[1,a-1]}) > (\frac{1}{2}+\alpha)^{\frac{\ell}{2}}] \quad \leq \quad 2\beta$$

for $\ell = b - a + 1$.

Proof. We say $i \in [m]$ is bad for $z \in \mathrm{support}(Z)$ if

$$\Pr_Z[Z_i = z_i | Z_{[1,i-1]} = z_{[1,i-1]}) > \frac{1}{2} + \alpha.$$

By assumption, for all i, $\Pr_Z[i \text{ is bad for } Z] \leq \beta$. Hence, using Markov,

$$\Pr_Z[\text{at least half of } i \in [a,b] \text{ are bad for } Z] \leq 2\beta$$

Whenever at most half of $i \in [a,b]$ are bad for z, $\Pr(Z_{[a,b]} = z_{[a,b]} | Z_{[1,a-1]} = z_{[1,a-1]}) \leq (\frac{1}{2}+\alpha)^{-\frac{\ell}{2}}$, and the lemma follows. □

We can now prove Thorem 5.

Proof. (Of Theorem 5) Suppose $Z = Z_1 \circ \ldots \circ Z_m$ is β–almost α–semi random source. That is Z is β–almost $((1,p),\ldots,(1,p))$ block-wise source for $2^{-p} = \frac{1}{2}+\alpha$. Partition the m bits into $b = O(\log^* m)$ blocks $W_1 \circ \ldots \circ W_b$ of length $n_1,\ldots,n_b$ as in the preceding section. Then by Lemmas 5.3 and 3.2 Z is $O(\beta b) = O(\beta\log^* m)$ close to a $(n_1,\ldots,n_b)$ block-wise source with $p' = \frac{p}{2}$. In particular, we can apply the extractor F of Theorem 7 with error $\beta\log^* m$. Thus,:

- The error is $\epsilon = O(\beta\log^* m)$.
- The number of truly random bits is $r = \mathrm{poly}(\log\frac{1}{p\epsilon})$.
- The number of output bits is at least $p'm + r - O(\log^* m(\log^* m + \log\epsilon^{-1}))$.

and we get Theorem 5. □

Acknowledgements

We thank Chris Umans, Ronen Shaltiel, and Oded Goldreich for helpful discussions and comments.

References

[AGHP92] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k–wise independent random variables. *Random Structures and Algorithms*, 3(3):289–303, 1992.

[CG88] B. Chor and O. Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, 1988.

[GHSZ00] V. Guruswami, J. Hastad, M. Sudan, and D. Zuckerman. Combinatorial bounds for list decoding. In *Proceedings of the 38th Annual Allerton Conference on Communication, Control, and Computing*, pages 603–612, 2000.

[GZ97] O. Goldreich and D. Zuckerman. Another proof that BPP $\subseteq$ PH (and more). Technical Report TR97-045, Electronic Colloquium on Computational Complexity, 1997.

[Hås96] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 627–636, 1996.

[ISW99] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Near-optimal conversion of hardness into pseudo-randomness. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 181–190, 1999.

[ISW00] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Extractors and pseudo-random generators with optimal seed length. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 1–10, 2000.

[MU01] E. Mossel and C. Umans. On the complexity of approximating the vc dimension. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity*, 2001.

[NN93] J. Naor and M. Naor. Small–bias probability spaces: Ef£cient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993.

[NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

[NZ96] N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.

[RRV99] R. Raz, O. Reingold, and S. Vadhan. Extracting all the randomness and reducing the error in Trevisan's extractors. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 149–158, 1999.

[RSW00] O. Reingold, R. Shaltiel, and A. Wigderson. Extracting randomness via repeated condensing. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 22–31, 2000.

[RTS00] J. Radhakrishnan and A. Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Discrete Mathematics*, 13(1):2–24, 2000.

[RZ98] A. Russell and D. Zuckerman. Perfect-information leader election in $\log^* n + O(1)$ rounds. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 576–583, 1998. Final version to appear in the Journal of Computer and System Sciences.

[San87] M. Santha. On using deterministic functions in probabilistic algorithms. *Information and Computation*, 74(3):241–249, 1987.

[Sip88] M. Sipser. Expanders, randomness, or time vs. space. *Journal of Computer and System Sciences*, 36:379–383, 1988.

[STV99] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 537–546, 1999.

[SU01] R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, 2001.

[Sud97] M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13, 1997.

[SV86] M. Santha and U. V. Vazirani. Generating quasi-random sequences from semi-random sources. *Journal of Computer and System Sciences*, 33:75–87, 1986.

[SZ99] A. Srinivasan and D. Zuckerman. Computing with very weak random sources. *SIAM Journal on Computing*, 28:1433–1459, 1999.

[Tre99] L. Trevisan. Construction of extractors using pseudo-random generators. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 141–148, 1999.

[TS96] A. Ta-Shma. On extracting randomness from weak random sources. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 276–285, 1996.

[TSUZ01] A. Ta-Shma, C. Umans, and D. Zuckerman. Loss-less condensers, unbalanced expanders, and extractors. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 143–152, 2001.

[TSZ01] A. Ta-Shma and D. Zuckerman. Extractor codes. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 193–199, 2001.

[Uma99] C. Umans. Hardness of approximating Σ_2^p minimization problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 465–474, 1999.

[WZ99] A. Wigderson and D. Zuckerman. Expanders that beat the eigenvalue bound: Explicit construction and applications. *Combinatorica*, 19(1):125–138, 1999.

[Zuc90] D. Zuckerman. General weak random sources. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 534–543, 1990.

[Zuc96] D. Zuckerman. Simulating BPP using a general weak random source. *Algorithmica*, 16:367–391, 1996.

[Zuc97] D. Zuckerman. Randomness-optimal oblivious sampling. *Random Structures and Algorithms*, 11:345–367, 1997.