Simple Extractors for All Min-Entropies and a New Pseudo-Random Generator

Ronen Shaltiel *

Christopher Umans †

Abstract

We present a simple, self-contained extractor construction that produces good extractors for all min-entropies (min-entropy measures the amount of randomness contained in a weak random source). Our construction is algebraic and builds on a new polynomial-based approach introduced by Ta-Shma, Zuckerman, and Safra [37]. Using our improvements, we obtain, for example, an extractor with output length $m = k^{1-\delta}$ and seed length $O(\log n)$. This matches the parameters of Trevisan's breakthrough result [38] and additionally achieves those parameters for small *min-entropies k. Extending [38] to small k has been the focus of a sequence of recent works [15, 26, 35]. Our construction gives a much simpler and more direct solution to this problem.*

Applying similar ideas to the problem of building pseudo-random generators, we obtain a new pseudo-random generator construction that is not *based on the NW generator [21], and turns worst-case hardness* directly *into pseudo-randomness. The parameters of this generator match those in [16, 33] and in particular are strong enough to obtain a new proof that $P = BPP$ if E requires exponential size circuits.*

Essentially the same construction yields a hitting set generator with optimal *seed length that outputs $s^{\Omega(1)}$ bits when given a function that requires circuits of size s (for* any *s). This implies a hardness versus randomness tradeoff for RP and BPP that is optimal (up to polynomial factors), solving an open problem raised by [14]. Our generators can also be used to derandomize AM in a way that improves and extends the results of [4, 18, 20].*

1. Introduction

Extractors and pseudo-random generators are two fundamental constructs from derandomization theory; both address the problem of simulating randomized algorithms on inherently deterministic machines. Pseudo-random generators (PRGs) are algorithms that stretch a short seed of truly random bits into a long string of bits that are indistinguishable from truly random bits to all small circuits. PRGs imply the existence of hard functions and, in the absence of strong circuit lower bounds, are constructed using the assumption that hard functions exist. This paradigm was initiated by [7, 41]. Extractors (defined in [22]) are algorithms that use a short seed of truly random bits to transform a weak random source (which we imagine to be some physical source of crude randomness) into a long string of bits statistically indistinguishable from truly random bits.

The seed length and the output length are the two major parameters of interest for both extractors and PRGs. For PRGs, we hope to produce $s^{\Omega(1)}$ bits when given a function on $\log n$ bits which requires circuits of size s, while for extractors, we hope to produce k output bits from any n-bit long weak random source with min-entropy k. For both, we hope for a seed length of $O(\log n)$ so that we can enumerate all seeds in polynomial time.

A long line of research initiated by Nisan and Wigderson [21, 5, 12, 16, 33, 14, 15] has focused on constructing PRGs using the assumption that there exists a function family in $E = \text{DTIME}(2^{O(n)})$ which requires large circuits. Similarly, a long line of research [22, 43, 34, 31, 38, 25, 15, 26, 35] has focused on constructing extractors. The two areas were linked by Trevisan [38], who showed that Nisan-Wigderson-style PRGs, properly interpreted, are actually extractors.

Extractors have been used in a remarkable variety of settings beyond their intended application. These include complexity theory [30, 22, 10], algorithms [40], hardness of approximation [42, 39], distributed protocols [43, 27], and coding theory [36]. Similarly, PRGs have been used to prove statements that are, on their surface, unrelated to derandomization; see for example [18, 17, 19, 13]. Quite good parameters have been achieved for extractors and PRGs, but the array of applications (and potential applications) suggests that further optimization is still valuable.

The main contribution of this paper, however, is *not* optimizing parameters, but developing a new, simple, self-contained, and versatile construction of both extractors and PRGs that achieves good results for the whole range of parameters. Such a construction is important from a practi-

*Hebrew University, Jerusalem, Israel and Institute for Advanced Study, Princeton, NJ 08540. ronens@cs.huji.ac.il

†Microsoft Research, One Microsoft Way, Redmond, WA 98052. umans@microsoft.com

cal point of view, and also may advance the technical effort to construct extractors matching the lower bounds by introducing new ideas and stripping away the complicated recursions, iterations, and compositions that characterized much of the previous work. We build on a new technique introduced in [37] for building extractors, contributing a number of key ideas that allow it to be used for PRGs, and for extractors over a broad parameter range (the extractors in [37] only work when the min-entropy is quite high, and even then they lose quite a bit of the source randomness).

In terms of actual parameters, our extractor constructions do not break any records, but do match constructions that were considered breakthroughs only a few years ago. In the area of PRGs, we obtain some concrete improvements, solving open problems raised by [14] and [20]. Below, we outline our results in more detail:

- Our extractors (summarized in Table 1) all come from the same simple polynomial-based construction, which we tune to achieve a variety of parameters. Our first extractor achieves a relatively large output length, while retaining the asymptotically optimal seed length. This matches the seed length and output length of Trevisan's extractor (which requires $k = n^{\Omega(1)}$), but also works for small k, giving a more direct solution to the problem addressed by [15, 35]. Our second extractor achieves a seed length with near-optimal leading constant for all min-entropies. This extends the result of [37] which works only for large min-entropy. Our third extractor almost matches the output length of [26] (the current best construction for general k with seed length $O(\log n)$), and *simultaneously* achieves a seed length with near-optimal leading constant. Unlike the first two extractors it is not self-contained – it relies on another extractor construction from [37, 22].

- Prior to this paper, all known PRG constructions were based on the original NW PRG (with the exception of the Blum-Micali-Yao-style PRGs [7, 41], which are based on a stronger hardness assumption). Coming up with an alternate construction has long been an open problem. Our construction does *not* use the NW PRG, and is arguably simpler than previous constructions. In particular, there is no explicit hardness-amplification component: we transform worst-case hardness directly into pseudo-randomness. The parameters of our PRG match [16, 33] (that is, seed length $O(\log^2 n/\log s)$ and output length $s^{\Omega(1)}$) and consequently we obtain a new proof that $BPP = P$ if there exists a function family in E which requires exponential size circuits.

- A hitting set generator (HSG) is the one-sided variant of a PRG, and the canonical construct for derandomizing RP (instead of BPP). We give the first construction of an *optimal* HSG; that is, a HSG with seed length $O(\log n)$ that outputs $s^{\Omega(1)}$ bits when given a function over $\log n$ bits which requires size s circuits. Our construction is optimal in the sense made formal in [14]; namely, any construction which does significantly better also produces a harder function than the one initially supplied to it.

  Additionally, any conversion of hardness into pseudorandomness with a black-box proof (in the sense of [38]) yields extractors or dispersers, and therefore the unconditional lower bounds of [22, 23] apply. It should be noted, however, that we are more liberal in what we demand from optimal generators compared to optimal extractors. It does not make sense to optimize the leading constant in the seed length or the constant in the exponent of the output length as these are spoiled anyway when using generators for derandomization.

  An optimal HSG immediately gives an optimal hardness versus randomness tradeoff for RP. Using the result of [1] that HSGs suffice to derandomize BPP,[1] we obtain an optimal hardness versus randomness tradeoff for BPP. Specifically, we get the following tradeoff: if there exists a family of functions in E requiring circuits of size $s(n)$, then BPTIME$(t) \subseteq$ DTIME$(2^{O(s^{-1}(t^{O(1)}))})$. This completes a line of research begun in [14] and in particular improves the best previous result (due to [14, 15]) which obtained the weaker conclusion that BPTIME$(t) \subseteq$ DTIME$(2^{O(s^{-1}(t^{O(\log\log t)}))})$.

- Specific properties of our proof technique allow us to prove that both our PRG and HSG, with identical parameters, can be used to fool *non-deterministic circuits* when given a function that is hard for *single-valued-non-deterministic* circuits. This gives an optimal hardness versus randomness tradeoff for AM (the $AM \stackrel{?}{=} NP$ problem is the non-deterministic analog of the $BPP \stackrel{?}{=} P$ problem), and improves and extends [18] (which relied on a stronger hardness assumption) and [20] (which does not work for all hardnesses). Due to space limitations, we defer this result in its entirety to the final version.

While the constructions in [37] and the present work are simple, the proofs are more involved. The common thread in the proof techniques of [33], [37] and this work is the use of specific properties of error-correcting codes, and ideas for decoding them. In the Section 2 we describe the construction of [37] and our improvements at a high level.

2. Overview of the Technique

We first outline the relevant prior work on extractors in section 2.1, then our new ideas in section 2.2, and finally we describe the additional ideas needed to build PRGs and HSGs in section 2.3.

[1]In our setting it is also possible to use the much simpler "tournament of generators" from [14].

Additional randomness t	Output length m	which k	Reference
$O(\log n)$	$k^{1-\delta}$	$k > \log^{O(1/\delta)} n$	Corollary 1
$(1+\alpha)\log n$	$k^{\Omega(\alpha)}$	any k	Corollary 2
$(1+\alpha)\log n$	$k/(\log^{O(1/\alpha)} n)$	any k	Corollary 4

Table 1. extractors constructed in this paper ($\delta > 0$ is any constant and $\alpha < 1$).

2.1. Previous work

Trevisan [38] showed that a powerful proof technique that at first seems natural only for PRGS can in fact be used for extractors. The proof technique works by contradiction. One assumes that the extractor's output is not close to uniform and therefore does not pass some prediction test; that is, there exists a function f (the *predictor*) whose value is with non-negligible probability the i-th symbol output by the extractor when it is given the first $i-1$ symbols. A contradiction is derived by giving a "reconstruction procedure" based on the predictor f. This procedure is able to "reconstruct" the string sampled from the weak random source from a short advice string, for a large fraction of source strings. If the source has large min-entropy, this is a contradiction, because it implies that a large number of source strings have short descriptions (and there are only a few short descriptions).

Ta-Shma, Zuckerman and Safra [37] proposed a new extractor construction based on polynomials that uses this proof technique in a new way. Their construction is very simple. One thinks of the string sampled from the weak random source as a low-degree multivariate polynomial p over a finite field and of the seed as a random evaluation point. The extractor computes m "successive" points (starting from the random evaluation point) and outputs the evaluation of the polynomial at these points. The meaning of "successive" is purposely left a bit vague in this presentation and we will elaborate on it later.

Their proof describes a reconstruction procedure that attempts to reconstruct a source string (which is a polynomial p) using a predictor f and a short advice string which contains the value of p at a fixed small set of points. The basic idea is to use $i-1$ successive points from within this set to predict the value of p at the "next" point (using f). Now the predictor can be "advanced" by one step and use $i-2$ successive points plus the one just predicted to predict the value of p at the next point, and so on, until all of p is reconstructed. However, there are two complications.

First, the predictor is only correct with a small advantage over random guessing. To overcome this, at each step the predictor is used to predict in parallel all points along a random line L. Because points on a random line are pairwise independent, with high probability the predictor is correct on the same fraction of points in L as its total advantage over the whole space. The collection of values along the line can now be error-corrected since p restricted to L is a low-degree univariate polynomial.

Second, the relative number of errors is so large that unique decoding is impossible, and one must use list-decoding to obtain a small number of possible options, one of which is correct. To pin down the correct one, the advice string includes the evaluation of p at a random point on L, which with high probability agrees with *only* the correct item in the list.

We note that in this summary the extractor and the predictor output field elements instead of bits; however there is an easy reduction of the binary case to this one (see Section 3.4).

2.2. Improvements of this paper

The key to improving the quality of the [37] extractor is reducing the length of the advice string. To do this one has to decrease the degree of p, which increases the dimension. But, in [37] increasing the dimension *increases* the length of the advice string. Our improvements enable us to increase the number of dimensions without this deleterious effect.

An algebraic approach. What does "successive" mean? In [37] (say in two dimensions over finite field F), the successor of (a_1, a_2) is (a_1+1, a_2). The rational is that starting from any point (a_1, a_2) and taking successive steps one covers the line $F \times a_2$. Doing this for all a_2's covers the plane. However, this geometric approach succumbs to the "curse of dimensionality" as the dimension is increased.

The main source of our improvement comes from taking an algebraic instead of a geometric approach. The polynomials we wish to learn are defined over the vector-space F^d. Our insight is to view this space as an extension field of F. The multiplicative group of this field is cyclic and has generator g, and for us, the successor of an element is obtained by *multiplication* by g. This indeed has the essential property that by repeatedly taking successors, we cover the whole space. We also use critically that multiplication by g corresponds to a *linear transform* in the vector-space, so that lines get mapped to lines. Replacing the geometric approach by an algebraic approach avoids the geometric structure of F^d and now the dimension does not come into play.

Curves instead of lines. In [37], each prediction step fails with probability inversely proportional to the size of the field F, and they use a union bound to argue that *no* prediction steps fails during the entire reconstruction process.

Increasing the number of dimensions decreases the number of points in a line and means that it takes many more prediction steps to reconstruct the whole space and therefore many more events are in the union bound. It also decreases the field size which increases the failure probability of each individual event. Together these effects overwhelm us.

We overcome this by predicting along *degree r curves* instead of lines. Since the collection of points on such curves is r-wise independent, we can use higher moment tail inequalities to argue that the failure probability of each prediction step is *exponentially small* in r, which permits us to use the union bound even for our much larger collection of events. Using curves instead of lines is also necessary for the improvement described next.

Interleaved reconstruction procedures. There's an additional cost to using more prediction steps. Recall that in [37], the advice string must include the value of the polynomial at a random point on the line, *for each prediction step*. Having to include these will blow up the size of the advice string.

To overcome this problem we run two "interleaved" reconstruction procedures. Each uses its own random curve but we arrange it so that the two curves intersect at a few random points. The two reconstruction procedures works on their own. However, when one needs the value of the polynomial at a random point on its curve, it can use the value *already calculated* by the other reconstruction procedure instead of relying on the advice string. Thus, no additional information is required in the advice string beyond the values needed to get the two interleaved reconstruction procedures started.

We stress that almost all of what we have described in the preceding subsections relates to the *proof* that our construction is indeed an extractor. The extractor construction itself remains very simple.

2.3. Constructing PRGs and HSGs

Since the proof technique outlined above was originally applied to PRG constructions, it is easy to adapt to that setting. To convert our extractor into a PRG, we fix the "source string" x to be the truth table of a hard function, use the seed as before to pick an evaluation point, and output m successive evaluations. If this in *not* a PRG, then there is an *efficient* predictor f, and we wish to use f to produce a *small* circuit C that computes the function $x(j) = x_j$, contradicting the hardness of x.

Two things contribute to the size of C: the length of the advice string (which must be hardwired into C), and the number of prediction steps (since each step invokes f and requires computation to perform the list-decoding). Because the advice string must be small, the improvements we get over [37] in this area are essential.

However, the natural adaptation of our extractor into a PRG suffers from an inherent problem of the method of [37]: It is highly sequential. Specifically, computing x_j at a position j that is "far away" from the points specified by the advice string (one that takes many successive prediction steps to get to) takes too many steps and makes C too large to derive a contradiction.

The problem is that we can't have a "successor function" defined over F^d in such a way that very few applications of the function can get to *every* point from a fixed starting location. A helpful idea is to allow *several* such functions so that short sequences of applications of the different functions can reach every point. To achieve this we will take the first successor function to be the one we used for the extractor, the second successor function to be *m applications* of the first, the third to be m^2 applications of the first and so on. By first taking a few small strides, then a few larger strides, then a few even larger strides, etc., we can reach every point in a small number of steps.

Each one of these successor functions corresponds to a construction very similar to our extractor construction. We show that *at least one* of these constructions must be a PRG, since if none of them are, then we have predictors for all of them which gives us the predictors with differing strides needed to contradict the hardness of x.

By running all of these "candidate" PRGs with independent seeds, and XOR-ing their output, we obtain the desired single PRG. Because we need so few candidate generators (a *constant* number if x has exponential hardness), the seed length of the XORed generator is still relatively short, and we match the parameters of [33]. By taking the *union* of these candidate PRGs, we obtain the HSG with optimal seed length described in the introduction.

3. Extractors

We begin with some standard definitions. Whenever we use U_n, we mean the random variable that is uniform on $\{0,1\}^n$. Two distributions on the same domain are *ϵ-close* if the statistical distance between them is at most ϵ; i.e., the probabilities they give to any event differ by at most ϵ. A random variable X has min-entropy at least k if $\Pr[X = x] \leq 2^{-k}$ for all x; formally:

Definition 1 (min-entropy) *The* min-entropy *of a random variable X over $\{0,1\}^n$, written $H_\infty(X)$, is defined as* $H_\infty(X) = \min_{x\in\{0,1\}^n} -\log_2 \Pr[X = x]$.

Definition 2 (extractor) *A (k,ϵ)* extractor *is a function $E : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ such that for all random variables X with $H_\infty(X) \geq k$:*

$$E(X, U_t) \text{ is } \epsilon\text{-close to } U_m. \tag{1}$$

To show that (1) above holds, it is sufficient to prove (2):

$$\forall \;\; 1 \leq i \leq m \text{ and all functions } f : \{0,1\}^{i-1} \to \{0,1\},$$
$$\Pr[f(E(X,U_t)_{1\ldots i-1}) = E(X,U_t)_i] \leq \frac{1}{2} + \frac{\epsilon}{m} \tag{2}$$

and indeed the proofs for many recent extractor constructions follow this route. Property (2) requires that each successive bit of output be "unpredictable" based on the previous bits.

As our construction of extractors is algebraic, we will be working over finite fields with q elements. It will therefore be useful to define a "q-ary" extractor. Such an extractor is required to satisfy an unpredictability property analogous to (2); however, in the larger field we allow the prediction function f to output a small list of possible next elements, instead of just one.

Definition 3 (q-ary extractor) *Let F be the field with q elements. A (k, ρ) q-ary extractor is a function $E : \{0,1\}^n \times \{0,1\}^t \to F^m$ such the that for all random variables X with $H_\infty(X) \geq k$:*

$$\forall \ 1 \leq i \leq m \text{ and all functions } f : F^{i-1} \to F^{(\rho^{-2})},$$
$$\Pr[\exists j \ f(E(X,U_t)_{1\ldots i-1})_j = E(X,U_t)_i] \leq \rho. \quad (3)$$

In [37] it was shown how to transform q-ary extractors into (regular) extractors, and the details are given in section 3.4. This allows us to focus on building good q-ary extractors.

We need the following list-decoding bound due to [32]:

Lemma 1 ([32]) *Given n distinct pairs (x_i, y_i) in field F and parameters k and d, with $k > \sqrt{2dn}$, there are at most $2n/k$ degree d polynomials g such that $g(x_i) = y_i$ for at least k pairs. Furthermore, a list of all such polynomials can be computed in time poly$(n, \log|F|)$.*

We also need the following tail inequality for t-wise independent random variables:

Lemma 2 ([6]) *Let $t > 4$ be an even integer. Suppose $X_1, X_2, \ldots, X_n$ are t-wise independent random variables taking values in $[0,1]$. Let $X = \sum X_i$, $\mu = E[X]$ and $A > 0$. Then:*

$$\Pr[|X - \mu| \geq A] \leq 8 \cdot \left(\frac{t\mu + t^2}{A^2}\right)^{t/2}$$

3.1. The construction

Our construction is very simple. Like [37] our first step is to encode the string x from the weak random source with a q-ary Reed-Muller (d-variate polynomial) code. The coordinates of such a code are in one-to-one correspondence with the vectors in the vector-space F^d. Our q-ary extractor uses its truly random bits to pick a random $\vec{v} \in F^d$. It outputs the $\vec{v}$-th symbol of the encoded string and $m-1$ successive symbols. The successor of $\vec{v}$, is $A\vec{v}$, where A is a special $d \times d$ matrix, on which we elaborate below.

More formally, let F be the field with q elements, and fix the dimension d. Let h be the smallest integer such that

$$\binom{h+d}{d} \geq \frac{n}{\log q}. \quad (4)$$

For $x \in \{0,1\}^n$, let $\hat{x}$ denote the d-variate polynomial of total degree $h-1$ whose coefficients are specified by x. Such a polynomial has $\binom{h+d}{d}$ coefficients, so (4) implies that distinct x give rise to distinct $\hat{x}$.

The important property that we require of matrix A is that it "generates" $F^d \setminus \{0\}$. That is, for any non-zero vector $\vec{v} \in F^d$,

$$\{A^i\vec{v}\}_i = F^d \setminus \{0\}. \quad (5)$$

It is easy to show that such a matrix exists and can be found efficiently, using the algebraic structure of F^d:

Lemma 3 *Matrix A satisfying (5) exists and can be found in time $q^{O(d)}$.*

Proof: There is a natural isomorphism between the field $GF(q^d)$ and the vector-space F^d. Let g be a generator of the multiplicative group of $GF(q^d)$ (which is cyclic). Multiplication in the field corresponds to linear transforms in the vector-space, so the linear transform A corresponding to multiplication by g satisfies (5). We can find g by exhaustive search[2] in time $q^{O(d)}$. •

We can now define our q-ary extractor $E : \{0,1\}^n \times \{0,1\}^{d\log q} \to F^m$. We interpret the second input of the q-ary extractor as a vector $\vec{v} \in F^d$.

$$E(x, \vec{v}) = \hat{x}(A^1\vec{v}) \circ \hat{x}(A^2\vec{v}) \circ \cdots \circ \hat{x}(A^m\vec{v}) \quad (6)$$

Our main theorem is the following:

Theorem 1 (extractor main) *For every n, q, d and h satisfying (4), E is a (k, ρ) q-ary extractor, provided that*

1. *$k > \Omega(mhd \log q) + \log\left(\frac{1}{\rho}\right)$ and $q > \Omega\left(\frac{(hd)^2}{\rho^4}\right)$, or*
2. *$k > \Omega(mhd \log^2 q) + \log\left(\frac{1}{\rho}\right)$ and $q > \Omega\left(\frac{hd\log q}{\rho^4}\right)$.*

In section 3.4 we explain how to choose parameters to derive the extractors in Table 1 from the main theorem.

3.2. The reconstruction proof paradigm

To prove that E is a q-ary extractor, we use ideas that originated in [38] and are refined in [37], which one might label the "reconstruction proof paradigm".

Fixing a random variable X over $\{0,1\}^n$ with $H_\infty(X) \geq k$, we use a function f that violates property (3) to describe a "reconstruction function" $R : \{0,1\}^a \to \{0,1\}^n$. Function R has oracle access to f, and we design R to have the following property:

$$\Pr_{x \leftarrow X}[\exists z \ \ R^f(z) = x] \geq p. \quad (7)$$

[2] More precise bounds, and significant improvements in certain cases can be found in [28] and [29].

Intuitively, this implies that for a large fraction of strings x in the support of X, there is a short "advice string" z from which we can reconstruct x. Typically the advice string consists of small portions of an encoded version of x, and R uses ideas from decoding error-correcting codes to "reconstruct" x from this partial information. There are at most 2^a strings in $\{0,1\}^n$ which can be described using an a-bit long advice string. The total weight of these strings sampled according to X is at most $2^a \cdot 2^{-k}$. Thus, we get a contradiction if $2^{a-k} < p$.

Our main task is to use the "next-element predictor" function f in a clever way to allow many strings x in the support of X to be reconstructed from very short advice strings.

To obtain R, we use essentially the framework of [37], which we now describe more precisely. We use f in a sequence of "prediction steps." In each prediction step we learn the value of $\hat{x}$ evaluated at some points in F^d. In the end, we know $\hat{x}$ evaluated at *all* points in F^d, and we can interpolate to recover the coefficients of $\hat{x}$ and finally x.

Step i requires the evaluations of $\hat{x}$ at a set of "known points" K_i and uses f to predict $\hat{x}$ at a set of points P_i. We will prime the pump for the first t steps (for some small t) by supplying an advice string that can be used to obtain $\hat{x}$ evaluated at $K_1, K_2 \ldots K_t$. After that, we require that:

$$\forall i \geq t \quad P_i \cup \bigcup_{j=1}^{i} K_j \supseteq K_{i+1}. \tag{8}$$

In the next section we will use a probabilistic argument to show that there *exists* a sequence of prediction steps that works for many x in the support of X, and requires very few bits of advice to get started.

3.3. Proof of the main theorem

As outlined above, we fix a random variable X with $H_\infty(X) \geq k$ and a predictor function $f : F^{i^*-1} \to F^{(\rho^{-2})}$ violating property (3). The crux of the proof is the following lemma, which gives, with high probability, a prediction sequence that succeeds for every x on which the predictor gains an advantage.

Lemma 4 *Let n, q, d, and h be as in the statement of Theorem 1 (1) (resp. Theorem 1 (2)). There exists a probabilistic function $R : \{0,1\}^a \to \{0,1\}^n$ with $a = O(mhd \log q)$ (resp. $a = O(mhd \log^2 q)$) such that for every x on which*

$$\Pr_y[\exists j \ f(E(x,y)_{1\ldots i^*-1})_j = E(x,y)_{i^*}] > \rho/2,$$

the following holds (the probability is over the random coins of R):

$$\Pr[\exists z \quad R^f(z) = x] \geq 1/2.$$

Proof: We begin by describing the sets of points P_i that we will predict in each step in the prediction sequence. These will be low-degree curves in the space F^d, and will be chosen using the coin tosses of R.

Set $r = \Theta(d)$ (resp. $r = \Theta(d \log q)$), and pick $2r$ random points $\vec{y_1}, \vec{y_2}, \ldots \vec{y_{2r}}$ from F^d, and $2r$ values $t_1, t_2, \ldots t_{2r}$ from F. Define the degree $2r-1$ polynomial $p_1 : F \to F^d$ so that $p_1(t_i) = \vec{y_i}$, for $i = 1, 2, \ldots, 2r$. Define the degree $2r-1$ polynomial $p_2 : F \to F^d$ so that $p_2(t_i) = A\vec{y_i}$ for $i = 1 \ldots r$ and $p_2(t_i) = \vec{y_i}$ for $i = r+1 \ldots 2r$. We define $P_1 = \{p_1(z)\}_{z \in F}$ and $P_2 = \{p_2(z)\}_{z \in F}$. For $i \geq 1$ we define $P_{2i+1} = AP_{2i-1}$ and $P_{2i+2} = AP_{2i}$.

Using the fact that A is a non-singular linear transform we have for all $i \geq 1$:

- The set P_i is a $2r$-wise independent collection of points.
- P_i and P_{i+1} intersect at r random points.
- The polynomial $\hat{x}$ restricted to P_i (denoted $x_{|P_i}$) is a univariate polynomial of degree at most $2hr$.

We will set up the reconstruction function by supplying it with an advice string which will allow it to compute the evaluation of $\hat{x}$ on the points $P_1, \cdots, P_{2i^*-2}$. This will be done by giving as advice the $2hr$ coefficients of $\hat{x}|_{P_i}$ for $1 \leq i \leq 2i^* - 2$. Thus, the length of the advice string is at most $2mhr \log q \leq a$ as stated.

When given the evaluation of $\hat{x}$ at a series of points $A\vec{v}, A^2\vec{v}, \cdots, A^{i^*-1}\vec{v}$, we use the predictor function f to "attempt to predict" $\hat{x}(A^{i^*}\vec{v})$. Therefore, for all $j > 0$, given points:

$$\begin{aligned} &AP_j \cup A^2 P_j \cup \cdots \cup A^{i^*-1} P_j \\ &= P_{j+2} \cup P_{j+4} \cup \cdots \cup P_{j+2i^*-2} = K_{j+2i^*} \end{aligned}$$

we can use f to predict points $A^{i^*} P_j = P_{j+2i^*}$. Notice that our P_is and K_is satisfy (8) for $i > 2i^* - 2$.

So, in prediction step $i > 2i^* - 2$, we predict $\hat{x}$ on points P_i by invoking f once for each $\vec{z} \in P_i$. After doing that, we have a set of predicted values of size ρ^{-2} at each point $\vec{z} \in P_i$. We now show that with high probability these predicted values and the values of P_{i-1} completely determine the value of P_i.

Claim 1 *With probability at least $1 - 1/8q^d$ over the coin tosses of R:*

$$\Pr_{\vec{z} \in P_i}[\exists j \ f(\hat{x}(A^{-i^*+1}\vec{z}), \ldots, \hat{x}(A^{-1}\vec{z}))_j = \hat{x}(\vec{z})] \geq \rho/4.$$

Proof: We use that P_i is a $2r$-wise independent collection of points, and apply Lemma 2. •

Therefore, with high probability, we have $(\rho/4)q$ "good" evaluations that agree with the degree $2rh$ univariate polynomial $\hat{x}_{|P_i}$, out of a total of at most $\rho^{-2}q$. By our choice of q, Lemma 1 applies, and we conclude that at most $8\rho^{-3}$ degree $2rh$ polynomials agree on this number of points.

Now, P_i intersects P_{i-1} at r random positions, and we know the evaluation of $\hat{x}$ at the points in P_{i-1}. Two degree $2rh$ polynomials can agree on at most $2rh/q$ fraction of their points, so the probability that an "incorrect" polynomial from among our candidates agrees on all r random points is at most:

$$(8\rho^{-3})\left(\frac{2rh}{q}\right)^r < \frac{1}{8q^d}$$

where the final inequality holds by our choice of r and q.

So, with probability at least $1-\frac{1}{4q^d}$ over the random coins of R, we learn points P_i successfully.

After $2q^d$ prediction steps, we have learned $\hat{x}$ on $F^d\backslash\{\vec{0}\}$, by property (5) of matrix A. By the union bound, the probability that every step of this reconstruction is successful is at least $1/2$. •

Proceeding with the proof of the main theorem, we know that

$$\Pr_{x\leftarrow X,y}[\exists j\ f(E(x,y)_{1\ldots i^*-1})_j = E(x,y)_{i^*}] > \rho.$$

By an averaging argument,

$$\Pr_{x\leftarrow X}[\Pr_{y}[\exists j\ f(E(x,y)_{1\ldots i^*-1})_j = E(x,y)_{i^*}] \geq \rho/2] > \rho/2.$$

Therefore, there must be a fixing of the coin tosses of R for which we have:

$$\Pr_{x\leftarrow X}[\exists z \quad R^f(z) = x] > \frac{1}{2}\cdot\rho/2 = \rho/4.$$

Following the outline in the previous section, this implies that $\exp(O(mdh\log q) - k) \geq \rho/4$ (resp. $\exp(O(mdh\log^2 q) - k) \geq \rho/4$), which contradicts the lower bound on k in the statement of the theorem. This completes the proof of the main theorem.

3.4. From q-ary extractors to (regular) extractors

We are now left with the task of converting a q-ary extractor into a regular one. The standard way to go about is to use error correcting codes. The transformation described in the next easy lemma is essentially the information-theoretic analog of the hard-core bit constructions of Goldreich-Levin [8] and Impagliazzo.

Lemma 5 ([37]) *Let F be the field with q elements and let $C : \{0,1\}^{\bar{k}=\log q} \to \{0,1\}^{\bar{n}}$ be a binary error-correcting code with relative distance at least $1/2 - O(\rho^2)$. If $E : \{0,1\}^n \times \{0,1\}^t \to F^m$ is a $(k, O(\rho))$ q-ary extractor, then $E' : \{0,1\}^n \times \{0,1\}^{t+\log\bar{n}} \to \{0,1\}^m$ defined by:*

$$E'(x;(y,j)) = C(E(x;y)_1)_j \circ \cdots \circ C(E(x;y)_m)_j$$

is a $(k, \rho m)$ extractor.

There are explicit constructions of codes with the required minimum distance and short blocklength $\bar{n} = (\frac{\log q}{\rho})^{O(1)}$ (see, e.g., [11]). Thus, this transformation has minimal effect on the seed length. Even relatively simple codes, like a Reed-Solomon code concatenated with a Hadamard code yield the desired parameters. If we are not optimizing constants in the seed length, we can even afford to use perhaps the simplest binary code, the Hadamard code[3], which has relative distance 1/2.

We now combine Lemma 5 with Theorem 1 to obtain some specific extractors with good parameters.

Corollary 1 *For all n, ϵ, constant $\delta > 0$, and $k > \log^{4/\delta} n$ our construction gives a (k,ϵ)-extractor with seed length $t = O\left(\log n + \frac{\log n}{\log k}\log\left(\frac{1}{\epsilon}\right)\right)$ and output length $m = k^{1-\delta}$.*

One of the advantages of the extractors constructed in [37] are that they optimize the leading constant in the seed length. By picking parameters appropriately, we achieve a similar result.

Corollary 2 *For all n, k, ϵ and δ, our construction gives a (k,ϵ)-extractor with seed length $t = (1 + O(\delta))\log n + O(\delta)\left(\frac{\log n}{\log m}\log\left(\frac{1}{\epsilon}\right)\right)$ and output length $m = k^{\Omega(\delta)}$.*

Notice that when $k = n^{\Omega(1)}$, the seed length of both of these extractors is $O(\log(n/\epsilon))$, which gives the optimal dependence on ϵ. Achieving this dependence was a focus of [25, 24], and we obtain it directly.

The quality of our extractors can be significantly improved at the cost of using a more complex transformation of q-ary extractors to (regular) extractors. [37] construct a special kind of extractors which extract randomness from the output of q-ary extractors.[4]

Theorem 2 *[37] Let F be the field with $q < 2^m$ elements. There is a polynomial time computable function*

$$B : F^m \times \{0,1\}^{O\left(\log\frac{\log q\cdot\log^* m}{\rho}\right)^{O(1)}} \to \{0,1\}^{\Omega(m\log(1/\rho))}$$

such that for any (k,ρ) q-ary extractor E, $E'(x;(y,j)) = B(E(x,y),j)$ is a $(k, O(\rho\log^* m)$-extractor

The main advantage of the new method of [37] is that we no longer have to pay the $1/m$ penalty of using a hybrid argument (implicit in (2)). In order to get a (k,ϵ)-extractor we can have $\rho = O(\epsilon/\log^* m)$ rather than $\rho = O(\epsilon/m)$.

Using theorem 2 on top of our construction we get the following extractor:

Corollary 3 *For all n, k and constant ϵ our construction gives a (k,ϵ)-extractor with seed length $t = O(\log n)$ and output length $m = k/(\log n)^{O(1)}$.*

[3]The Hadamard encoding of a $\log q$-bit string x is $C(x) = \{\sum x_i y_i \bmod 2\}_{y\in\{0,1\}^{\log q}}$.

[4][37] prove this for $q = 2$. However, their methods also work for larger q.

This almost matches the parameters of the current best explicit construction for general min-entropy [26], and is by far a simpler construction.

We can further reduce the seed length to $(1+\delta)\log n$ at the cost of extracting slightly fewer bits.

Corollary 4 *For all $n, k, \delta < 1$ and constant ϵ our construction gives a (k,ϵ)-extractor with seed length $t = (1+\delta)\log n$ and output length $m = k/(\log n)^{O(1/\delta)}$.*

4. A new pseudo-random generator

In this section we exploit the connection Trevisan noticed between extractors and PRG's "the other way": we build PRGs (and HSGs) using the ideas outlined in section 2.3. Throughout, we identify the string $x \in \{0,1\}$ with the function $x : \{0,1\}^{\log n} \to \{0,1\}$ by setting $x(i) = x_i$. We denote by $S(x)$ the size of the smallest circuit computing function x.

Definition 4 (PRG) *An ϵ-*PRG for size *s is a function $G : \{0,1\}^t \to \{0,1\}^m$ with the following property: $\forall\ 1 \le i \le m$ and all functions $f : \{0,1\}^{i-1} \to \{0,1\}^i$ with size s circuits*

$$\Pr[f(G(U_t)_{1\ldots i-1}) = G(U_t)_i] \le \frac{1}{2} + \frac{\epsilon}{m}. \tag{9}$$

Property (9) implies the more standard requirement that $|\Pr[C(G(U_t)) = 1] - \Pr[C(U_m) = 1]| \le \epsilon$ for all size $s - O(1)$ circuits C. As in Section 3 we can define the q-ary version of PRGs.

Definition 5 (q-ary PRG) *Let F be the field with q elements. A ρ-q-ary* PRG for size *s is a function $G : \{0,1\}^t \to F^m$ with the following property: $\forall\ \ 1 \le i \le m$ and all functions $f : F^{i-1} \to F^{(\rho^{-2})}$ with size s circuits*

$$\Pr[\exists j\ f(G(U_t)_{1\ldots i-1})_j = G(U_t)_i] \le \rho. \tag{10}$$

An analog of Lemma 5 also holds and allows us to transform $O(\rho)$-q-ary PRGs for size s into (regular) $m\rho$-PRGs for size not much smaller than s. The details are standard and are omitted because of space limitations. In the rest of this section we present a construction of a q-ary generator.

4.1. The construction

Our construction uses a hard function $x : \{0,1\}^{\log n} \to \{0,1\}$ to build a PRG $G_x : \{0,1\}^t \to \{0,1\}^m$. Like in the extractor setting, we first encode x into $\hat{x}$ using a multivariate polynomial code. Here however, instead of using standard Reed-Muller codes, we need to take $\hat{x}$ to be the *low-degree extension* of x, and we need to be careful how we do even that.

Just as before, the major parameters are the field size q, the dimension d and the degree h. As in the previous section we will think of F^d as both a vector-space and the extension field of F. However, we also need a new ingredient. We will require that F has a subfield F' of size h (which forces h to be a prime power, and q to be a power of h). We note that F'^d is a subset of F^d, and we have the following lemma which extends Lemma 3.

Lemma 6 *There exist invertible $d \times d$ matrices A and A' with entries from F which satisfy:*

- *For any non-zero vector $\vec{v} \in F^d$: $\{A^i\vec{v}\}_i = F^d \setminus \{0\}$*
- *For any non-zero vector $\vec{v} \in F'^d$ $\{A'^i\vec{v}\}_i = F'^d \setminus \{0\}$.*
- *$A' = A^p$ for $p = (q^d - 1)/(h^d - 1)$.*
- *A and A' can be found in time $q^{O(d)}$.*

Proof: There is a natural correspondence between F^d and $GF(q^d)$, and between F'^d and $GF(h^d) \subseteq GF(q^d)$. As in Lemma 3, let g be a generator of the multiplicative group of $GF(q^d)$. This group is cyclic of order $q^d - 1$, and therefore g^p generates the *unique* subgroup of order $h^d - 1$, the multiplicative group of $GF(h^d)$. We take A and $A' = A^p$ to be the linear transforms corresponding to multiplication by g and g^p, respectively.

•

We require(compare to (4)):

$$h^d > n \tag{11}$$

Now, in our low degree extension, we will "embed" x into a polynomial defined over the vector-space F^d. The embedding works as follows: we want $\hat{x}(A'^i\vec{1}) = x(i)$. Here, $\vec{1}$ is the all-ones vector, which is in $F'^d \subseteq F^d$ since $1 \in F' \subseteq F$, and which serves as a reference vector thoughout the construction. Since $h^d - 1 \ge n$, there are enough "slots" to embed all of x, and since A' generates $F'^d \setminus \{0\}$, we have embedded all of x in a d dimensional cube of size h^d. Therefore there exists a low degree polynomial $\hat{x}$ over F^d with degree h in each variable such that $\hat{x}(A'^i\vec{1}) = x(i)$. Note that in this section h denotes the degree in individual variables and the total degree is at most hd. The computation of $\hat{x}$ from x can be done in poly$(n, q^d) = q^{O(d)}$ time.

We now describe ℓ "candidate" PRGs, where $\ell = d\frac{\log q}{\log m}$. For $0 \le j < l$ we define functions $G_x^{(j)} : \{0,1\}^{d\log q} \to F^m$ as follows:

$$G_x^{(j)}(\vec{v}) = \hat{x}(A^{1\cdot m^j}\vec{v}) \circ \hat{x}(A^{2\cdot m^j}\vec{v}) \circ \cdots \circ \hat{x}(A^{m\cdot m^j}\vec{v}).$$

Note that each $G_x^{(j)}$ corresponds to using our q-ary extractor construction with the "successor function" A^{m^j}. Our main theorem will show that at least one of these functions is a q-ary PRG, provided x is a sufficiently hard predicate.

Theorem 3 (PRG main) *For every $n, d,$ and h satisfying (11), at least one $G_x^{(j)}$ is an ρ-q-ary PRG for size s, provided that $S(x) > poly(s, m, 1/\rho, q)$ and $q > \Omega(\rho^{-4} h d^2 \log^2 q)$. Furthermore, all the $G_x^{(j)}$s are computable in time $poly(q^d, n)$ with oracle access to x.*

Proof: (sketch) We assume that none of the $G_x^{(j)}$s are PRGs, which implies that we have ℓ efficient predictors $f^{(0)}, f^{(1)}, \ldots, f^{(\ell-1)}$, where predictor $f^{(j)}$ has "stride" m^j. From these, we wish to construct a small circuit that computes $x(i)$ when given i as input. There are two aspects of the proof with multiple predictors that do not follow the extractor proof.

First, we need to describe the sequence in which we will invoke the various predictors to "move" from the fixed points whose evaluations are supplied by the advice string to the point $A'^i\vec{1} = A^{ip}\vec{1}$, which contains the value of $x(i)$. Suppose the advice string contains $\hat{x}$ evaluated at points $A^a\vec{1}, A^{a+1}\vec{1}, \ldots, A^{a+m-1}\vec{1}$, for some fixed constant a. We will use the predictors in order of increasing stride, each for m^2 prediction steps, maintaining the following property: after using predictor j, we will have learned $\hat{x}$ at a point $A^w\vec{1}$ for which the integers w and ip agree on the lowest $j+1$ digits of their m-ary representations.

We illustrate the first two phases of this process. First, we use $f^{(0)}$ to predict points $A^{a+m}\vec{1}, A^{a+m+1}\vec{1}, \ldots A^{a+m+m^2}\vec{1}$. Some integer b between $a+m^2$ and $a+m^2+m$ agrees with the lowest order digit of the m-ary representation of ip. Next, we use $f^{(1)}$ to predict points $A^{b+m}\vec{1}, A^{b+2m}\vec{1}, \ldots A^{b+m+m^2}\vec{1}$, noticing the important fact that in the $f^{(0)}$ phase, we have learned the points needed to start up $f^{(1)}$. We continue in this manner until we reach our target point $A^{ip}\vec{1}$. The total number of steps is ℓm^2.

The second tricky part of the proof arises because of course we are actually using the predictors along *curves*, and we are doing *interleaved predicting*. To handle these complications, the two random curves p_1 and p_2 along which the interleaved reconstruction procedures operate must be chosen carefully. We require that:

- for all i, $A^i p_1$ is an r-wise independent collection of points, and
- for all i, $A^i p_2$ is an r-wise independent collection of points, and
- for all i, $A^i p_1$ intersects $A^i p_2$ at r random positions, and for all "strides" $0 \le j < \ell$, $A^{i+m^j} p_1$ intersects $A^i p_2$ at r random positions

By choosing the curves in a slightly more complicated way than in the extractor proof (we actually use ℓr degree curves), we can ensure that these properties hold.

The remaining details of the proof are similar to the extractor proof, and we defer them to the final version. •

We now convert each $G_x^{(j)}$ into a binary function ${G'_x}^{(j)} : \{0,1\}^{O(d \log q)} \to \{0,1\}^m$ using the computational analogue of Lemma 5.[5] It follows that at least one of the ${G'_x}^{(j)}$s is a (binary) PRG. Our PRG is obtained by "XOR-ing" the l candidates functions:

$$G_x(y_0, \cdots, y_{\ell-1}) = {G'_x}^{(0)}(y_0) \oplus \cdots \oplus {G'_x}^{(\ell-1)}(y_{\ell-1})$$

It is standard that "XOR-ing" many candidates where one of them is a PRG for size s indeed produces a PRG for size $s-m$; for a proof, see [15]. It follows that G_x is an $O(m\rho)$-PRG for size $s-m$, whenever the conditions in Theorem 3 hold. By choosing q, d and h appropriately, we match the parameters of [33]:

Corollary 5 *For any s, if there exists a function $g : \{0,1\}^{\log n} \to \{0,1\}$ which is computable in time $n^{O(1)}$ with $S(g) \ge s$ then there exists a $1/m$-PRG for size m with seed length $t = O(\log^2 n / \log s)$ and output length $m = s^{\Omega(1)}$. Furthermore, this generator can be computed in time $n^{O(1)}$.*

The most important implication of this corollary is a new proof of the Impagliazzo-Wigderson Theorem [16], which states that $BPP = P$ if there exist a function family in E that requires exponential size circuits.

4.2. Hitting set generators and an optimal hardness vs. randomness tradeoff

Hitting set generators are designed to derandomize RP.

Definition 6 (HSG) *An ϵ-HSG for size s is a function $H_x : \{0,1\}^t \to \{0,1\}^m$ such that for all functions $C : \{0,1\}^m \to \{0,1\}$ computable by size s circuits that have $\Pr_z[C(z) = 1] \ge \epsilon$, there exists some y for which $C(H_x(y)) = 1$.*

One of the ${G'_x}^{(j)}$'s is a PRG and therefore if we choose a random candidate, we will hit the "right one" with positive probability. We define:

$$H_x(y, j) = {G'_x}^{(j)}(y)$$

Very few bits (at most $O(\log\log n)$) are needed to choose j and the seed length hardly changes. It follows that H_x is an $O(m\rho)$-HSG for size s whenever the conditions of theorem 3 hold. By choosing q, d and h appropriately, we get an *optimal* hitting set generator.

Corollary 6 *For any s, if there exists a function $g : \{0,1\}^{\log n} \to \{0,1\}$ which is computable in time $n^{O(1)}$ with $S(g) \ge s$ then there exists a $1/m$-HSG for size m with seed length $t = O(\log n)$ and output length $m = s^{\Omega(1)}$. Furthermore, this generator can be computed in time $n^{O(1)}$.*

[5] Lemma 5 only requires a code with efficient encoding; here we need a code that also has efficient (list-)decoding.

Using [1] (see also [3, 9]), we extend the Impagliazzo-Wigderson Theorem [16] to any hardness assumption.

Corollary 7 *If there exists a function family $g = \{g_n\} \in E$ which require size $s(n)$ circuits, then BPTIME$(\ell) \subseteq$ DTIME$(2^{O(s^{-1}(\ell^{O(1)}))})$.*

Acknowledgments We thank Henry Cohn, Venkat Guruswami, Valentine Kabanets, Omer Reingold, Muli Safra, Amnon Ta-Shma, Salil Vadhan, Avi Wigderson and David Zuckerman for helpful discussions. We are especially grateful to the authors of [37] for explaining their result to us. We thank the conference referees for helpful comments.

References

[1] A. Andreev, A. Clementi, and J. Rolim. Hitting sets derandomize BPP. In *ICALP*, 1996. Final version in J. ACM.

[2] A. Andreev, A. Clementi, and J. Rolim. A new general derandomization method. *J. ACM*, 45(1), 1998.

[3] A. E. Andreev, A. E. F. Clementi, J. D. P. Rolim, and L. Trevisan. Weak random sources, hitting sets, and BPP simulations. In *38th FOCS*, pages 264–272, 1997.

[4] V. Arvind and J. Kobler. On resource-bounded measure and pseudorandomness. *17th International Conference on Foundations of Software Technology and Theoretival Computer Science*, 1997.

[5] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[6] M. Bellare and J. Rompel. Randomness-efficient oblivious sampling. In *35th FOCS*, 1994.

[7] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, Nov. 1984.

[8] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *21st STOC*, pages 25–32, 1989.

[9] O. Goldreich, S. Vadhan, and A. Wigderson. Simplified derandomization of BPP using a hitting set generator. Technical Report TR00-004, ECCC, January 2000.

[10] O. Goldreich and D. Zuckerman. Another proof that BPP $\subseteq$ PH (and more). Technical Report TR97-045, ECCC, 1997.

[11] V. Guruswami and M. Sudan. List decoding algorithms for certain concatenated codes. In *32nd STOC*, 2000.

[12] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *36th FOCS*, pages 538–545, 1995.

[13] R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. In *16th Computational Complexity*, pages 1–12, 2001.

[14] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Near-optimal conversion of hardness into pseudo-randomness. In *40th FOCS*, 1999.

[15] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Extractors and pseudo-randomn generators with optimal seed-length. In *32nd STOC*, 2000.

[16] R. Impagliazzo and A. Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *29th STOC*, pages 220–229, 1997.

[17] V. Kabanets. Easiness assumptions and hardness tests: Trading time for zero error. In *15th Computational Complexity*, pages 150–157, 2000.

[18] A. R. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. In *31st STOC*, 1999.

[19] C. J. Lu. Derandomizing arthur-merlin games under uniform assumptions. In *16th ISAAC*, pages 302–312, 2000.

[20] P. B. Miltersen and N. V. Vinodchandran. Derandomizing Arthur-Merlin games using hitting sets. In *40th FOCS*, 1999.

[21] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, Oct. 1994.

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

[23] J. Radhakrishnan and A. Ta-Shma. Tight bounds for depth-two superconcentrators. In *38th FOCS*, pages 585–594, 1997.

[24] R. Raz, O. Reingold, and S. Vadhan. Error reduction for extractors. In *40th FOCS*, 1999.

[25] R. Raz, O. Reingold, and S. Vadhan. Extracting all the randomness and reducing the error in trevisan's extractors. In *31st STOC*, pages 149–158, 1999.

[26] O. Reingold, R. Shaltiel, and A. Wigderson. Extracting randomness via repeated condensing. In *41st FOCS*, 2000.

[27] A. Russell and D. Zuckerman. Perfect-information leader election in $\log^* n + O(1)$ rounds. *J. Comput. Syst. Sci.* To appear. Preliminary version in 39th FOCS.

[28] V. Shoup. Searching for primitive roots in finite fields. *Mathematics of Computation*, 58:369–380, 1992.

[29] I. Shparlinski. On finding primitive roots in finite fields. *Theoretical Computer Science*, 157:273–275, 1996.

[30] M. Sipser. Expanders, randomness, or time versus space. *Journal of Computer and System Sciences*, 36, 1988.

[31] A. Srinivasan and D. Zuckerman. Computing with very weak random sources. *SIAM Journal on Computing*, 28(4):1433–1459, Aug. 1999.

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

[33] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the xor lemma. In *31st STOC*, 1999.

[34] A. Ta-Shma. On extracting randomness from weak random sources. In *28th STOC*, pages 276–285, 1996.

[35] A. Ta-Shma, C. Umans, and D. Zuckerman. Loss-less condensers, unbalanced expanders, and extractors. In *33rd STOC*, 2001.

[36] A. Ta-Shma and D. Zuckerman. Extractor codes. In *33rd STOC*, 2001.

[37] A. Ta-Shma, D. Zuckerman, and S. Safra. Extractors from Reed-Muller codes. These Proceedings, 2001.

[38] L. Trevisan. Construction of extractors using pseudorandom generators. In *31st STOC*, 1999.

[39] C. Umans. Hardness of approximating Σ_2^p minimization problems. In *40th FOCS*, pages 465–474, 1999.

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

[41] A. C. Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd FOCS*, pages 80–91, 1982.

[42] D. Zuckerman. On unapproximable versions of NP-complete problems. *SIAM J. Comput.*, 25:1293–1304, 1996.

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