Extracting all the Randomness and Reducing the Error in Trevisan's Extractors

Ran Raz* Omer Reingold† Salil Vadhan‡

Abstract

We give explicit constructions of extractors which work for a source of any min-entropy on strings of length n. The first construction extracts any constant fraction of the min-entropy using $O(\log^2 n)$ additional random bits. The second extracts all the min-entropy using $O(\log^3 n)$ additional random bits. Both of these constructions use fewer truly random bits than any previous construction which works for all min-entropies and extracts a constant fraction of the min-entropy. We then improve our second construction and show that we can reduce the entropy loss to $2\log(1/\varepsilon)+O(1)$ bits, while still using $O(\log^3 n)$ truly random bits (where entropy loss is defined as [(source min-entropy) + (# truly random bits used) − (# output bits)], and ε is the statistical difference from uniform achieved). This entropy loss is optimal up to a constant additive term.

Our extractors are obtained by observing that a weaker notion of "combinatorial design" suffices for the Nisan–Wigderson pseudorandom generator, which underlies the recent extractor of Trevisan. We give near-optimal constructions of such "weak designs" which achieve much better parameters than possible with the notion of designs used by Nisan–Wigderson and Trevisan.

We also show how to improve our constructions (and Trevisan's construction) when the required statistical difference from uniform distribution ε is relatively small. This improvement is obtained by using multilinear error correcting codes over finite fields, rather than the arbitrary error correcting codes used by Trevisan.

*Department of Applied Mathematics and Computer Science, Weizmann Institute, Rehovot, 76100 Israel. E-mail: ranraz@wisdom.weizmann.ac.il Work supported by an American-Israeli BSF grant 95-00238 and by ESPRIT working group RAND2.

†Department of Applied Mathematics and Computer Science, Weizmann Institute, Rehovot, 76100 Israel. E-mail: reingold@wisdom.weizmann.ac.il Research supported by a Clore Scholars award and an Eshkol Fellowship of the Israeli Ministry of Science and by ESPRIT working group RAND2.

‡MIT Laboratory for Computer Science. 545 Technology Square. Cambridge, MA 02139. USA. E-mail: salil@theory.lcs.mit.edu. URL: http://theory.lcs.mit.edu/~salil. Supported by a DOD/NDSEG fellowship and partially by DARPA grant DABT63-96-C-0018.

STOC '99 Atlanta GA USA

1 Introduction

Roughly speaking, an extractor is a function which extracts (almost) truly random bits from a weak random source, using a small number of additional random bits as a catalyst. A large body of work has focused on giving explicit constructions of extractors, as such constructions have a wide variety of applications. A recent breakthrough was made by Luca Trevisan [Tre98], who discovered that the Nisan–Wigderson pseudorandom generator [NW94], previously only used in a computational setting, could be used to construct extractors. For certain settings of the parameters, Trevisan's extractor is optimal and improves on previous constructions. More explicitly, Trevisan's extractor improves over previous constructions in the case of extracting a relatively small number of random bits (e.g., extracting $k^{1-\alpha}$ bits from source with "k bits of randomness", where $\alpha > 0$ is an arbitrarily small constant) with a relatively large statistical difference from uniform distribution (e.g., constant ε, where ε is the statistical difference from uniform distribution required from the output). However, when one wants to extract more than a small fraction of the randomness from the weak random source, or when one wants to achieve a small statistical difference from uniform distribution, Trevisan's extractor performs poorly (in that a large number of truly random "catalyst" bits are needed).

In this paper, we show that Trevisan's ideas can be used in a more general and efficient way. We present two new ideas that improve Trevisan's construction. The first idea allows one to extract more than a small fraction of the randomness from the weakly random source. In particular, the idea can be used to extract all of the randomness from the weak random source. This is accomplished by improving the combinatorial construction underlying the Nisan–Wigderson generator used in Trevisan's construction. Applying a result of Wigderson and Zuckerman [WZ95] to these extractors, we also obtain improved constructions of highly expanding graphs and superconcentrators.

The second idea improves Trevisan's construction in the case where the output bits are required to be of a relatively small statistical difference from uniform distribution. The two ideas can be combined, and the final outcome is a set of new extractors that use fewer truly random bits than any previous construction which extracts at least a constant fraction of the randomness from any weak random source.

Extractors. A distribution X on $\{0,1\}^n$ is said to have *min-entropy* k if for all $x \in \{0,1\}^n$, $\Pr[X = x] \leq 2^{-k}$. Think of this as saying that X has "k bits of randomness." A function $\text{EXT}: \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is called an (k,ε)-*extractor* if for every distribution X on $\{0,1\}^n$ of min-entropy k, the in-

duced distribution $\text{EXT}(X, U_d)$ on $\{0,1\}^m$ has statistical difference at most ε from uniform (where U_d is the uniform distribution on $\{0,1\}^d$). In other words, EXT extracts m (almost) truly random bits from a source with k bits of hidden randomness using d additional random bits as a catalyst. The goal is to explicitly construct extractors which minimize d (ideally, $d = O(\log(n/\varepsilon))$) while m is as close to k as possible.[1] *Dispersers* are the analogue of extractors for one-sided error; instead of inducing the uniform distribution, they simply hit all but a ε fraction of points in $\{0,1\}^m$ with nonzero probability.

Previous work. Dispersers were first defined by Sipser [Sip88] and extractors were first defined by Nisan and Zuckerman [NZ96]. Much of the motivation for research on extractors comes from work done on "somewhat random sources" [SV86, CG88, Vaz87b, VV85, Vaz84, Vaz87a]. There have been a number of papers giving explicit constructions of dispersers and extractors, with a steady improvement in the parameters [Zuc96, NZ96, WZ95, GW97, SZ98, SSZ98, NT98, Zuc97, TS98b, Tre98]. Most of the work on extractors is based on techniques such as k-wise independence, the Leftover hash lemma [ILL89], and various forms of composition. A new approach to constructing extractors was recently initiated by Trevisan [Tre98], who discovered that the Nisan–Wigderson pseudorandom generator [NW94] could be used to construct extractors.

Explicit constructions of extractors and dispersers have a wide variety of applications, including simulating randomized algorithms with weak random sources [Zuc96]; constructing oblivious samplers [Zuc97]; constructive leader election [Zuc97]; randomness-efficient error reduction in randomized algorithms and interactive proofs [Zuc97]; explicit constructions of expander graphs, superconcentrators, and sorting networks [WZ95]; hardness of approximation [Zuc96]; pseudorandom generators for space-bounded computation [NZ96, RR99]; derandomizing BPP under circuit complexity assumptions [ACR97, STV98]; and other problems in complexity theory [Sip88, GZ97].

For a detailed survey of previous work on extractors and their applications, see [NT98].

Our results. The first family of extractors constructed in this paper are given in the following theorem:

Theorem 1 *For every n, k, m, and ε, such that $m \leq k \leq n$, there are explicit (k,ε)-extractors* $\text{EXT}: \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *with*

1. $d = O\left(\frac{\log^2(n/\varepsilon)}{\log(k/m)}\right)$, *or*
2. $d = O\left(\log^2(n/\varepsilon) \cdot \log(1/\gamma)\right)$, *where* $1+\gamma = k/(m-1)$, *and* $1/m \leq \gamma < 1/2$.

In particular, using the second extractor with $k = m$, we can extract all of the min-entropy of the source using

$$O\left(\log^2(n/\varepsilon) \cdot \log k\right)$$

additional random bits. (If ε is constant then this is just $O(\log^2 n \cdot \log k)$ additional random bits). Using the first extractor with k/m constant, we can extract any constant fraction of the min-entropy of the source using

$$O\left(\log^2(n/\varepsilon)\right)$$

additional random bits. (If ε is constant then this is just $O(\log^2 n)$ additional random bits).

[1] Actually, since the extractor is fed d truly random bits in addition to the k bits of hidden randomness, one can hope to have m be close to $k+d$. This will be discussed in more detail under the heading "Entropy loss."

An undesirable feature of the extractors in Theorem 1 (and the extractor of Trevisan [Tre98]) is that the number of truly random bits depends quadratically on $\log(1/\varepsilon)$. In (nonconstructive) optimal extractors and even some previous constructions (discussed later), this dependence is linear. Indeed, some applications of extractors, such as [RR99], require a linear dependence. In our second theorem, we improve our extractors to have a linear dependence on $\log(1/\varepsilon)$.

Theorem 2 *For every n, k, m, and ε, such that $m \leq k \leq n$, there are explicit (k,ε)-extractors* $\text{EXT}: \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *with*

1. $d = O\left(\frac{\log^2 n \cdot \log(1/\varepsilon)}{\log(k/m)}\right)$, *or*
2. $d = O\left(\log^2 n \cdot \log(1/\gamma) \cdot \log(1/\varepsilon)\right)$, *where* $1+\gamma = k/(m-1)$, *and* $1/m \leq \gamma < 1/2$.

Thus, in all cases, the $\log^2(n/\varepsilon)$ in Theorem 1 has been replaced with $\log^2 n \cdot \log(1/\varepsilon)$, which is an improvement when ε is relatively small. One case of note is when we want to extract $m = k^{1-\alpha}$ bits from a source of min-entropy $k \geq n^\alpha$, for an arbitrarily small constant $\alpha > 0$. This is the case in which Trevisan's extractor performs best, using $d = O(\log^2(n/\varepsilon)/\log n)$ truly random bits (which is $O(\log n)$ for constant $\varepsilon \geq 1/\text{poly}(n)$). In this case, Theorem 2 gives

$$d = O\left(\log n \cdot \log(1/\varepsilon)\right),$$

which is an improvement for small ε.

A summary of our results is given in Figure 1. A comparison with the best previous constructions is given in Figure 2. Trevisan's construction [Tre98] uses only $O(\log^2(n/\varepsilon)/\log k)$ truly random bits but extracts only a small fraction ($k^{1-\alpha}$) of the source min-entropy. The best previous construction that extracts all of the source min-entropy was given by Ta-Shma [NT98] and used $O(\log^9 n \cdot \log(1/\varepsilon))$ truly random bits.[2] Our extractors use more truly random bits than the extractor of [Zuc97] and the disperser of [TS98b], but our extractors have the advantage that they work for any min-entropy (unlike [Zuc97]), and are extractors rather than dispersers (unlike [TS98b]). The disadvantage of the extractors of [GW97] described in Figure 2 is that they only use a small number of truly random bits when the source min-entropy k is very close to the input length n (e.g., $k = n - \text{polylog}(n)$). There are also extractors given in [GW97, SZ98] which extract all of the min-entropy, but these use a small number of truly random bits only when the source min-entropy is very small (e.g., $k = \text{polylog}(n)$), and these extractors are better discussed later in the context of entropy loss.

Plugging the second extractor of Theorem 1 into a construction of [WZ95] (see also [NT98]) immediately yields the following type of expander graphs:

Corollary 3 *For every N and $K \leq N$, there is an explicitly constructible[3] graph on N nodes with degree*

$$(N/K) \cdot 2^{O((\log\log N)^2 (\log\log K))}$$

such that every two disjoint sets of vertices of size at least K have an edge between them.

[2] In [NT98], the number of truly random bits used by the extractor is given as $d = \text{polylog}\, n$, a polynomial of unspecified degree in $\log n$. Ta-Shma [TS98a] estimates the degree of this polynomial to be 9.

[3] By explicitly constructible, we mean that, given N and K, the graph can be constructed deterministically in time $\text{poly}(N)$.

reference	min-entropy k	output length m	additional randomness d	type
Thm. 1	any k	$m = (1-\alpha)k$	$d = O(\log^2(n/\varepsilon))$	extractor
Thm. 1	any k	$m = k$	$d = O(\log^2(n/\varepsilon) \cdot \log k)$	extractor
Thm. 2	any k	$m = k^{1-\alpha}$	$d = O(\log^2 n \cdot \log(1/\varepsilon)/\log k)$	extractor
Thm. 2	any k	$m = (1-\alpha)k$	$d = O(\log^2 n \cdot \log(1/\varepsilon))$	extractor
Thm. 2	any k	$m = k$	$d = O(\log^2 n \cdot \log(1/\varepsilon) \cdot \log k)$	extractor

Above, α is an arbitrarily small constant.

Figure 1: Summary of our constructions

reference	min-entropy k	output length m	additional randomness d	type
[GW97]	any k	$m = k$	$d = O(n - k + \log(1/\varepsilon))$	extractor
[Zuc97]	$k = \Omega(n)$	$m = (1-\alpha)k$	$d = O(\log(n/\varepsilon))$	extractor
[NT98]	any k	$m = k$	$d = O(\log^9 n \cdot \log(1/\varepsilon))$	extractor
[TS98b]	any k	$m = k - \text{polylog}(n)$	$d = O(\log(n/\varepsilon))$	disperser
[Tre98]	any k	$m = k^{1-\alpha}$	$d = O(\log^2(n/\varepsilon)/\log k)$	extractor
ultimate goal	any k	$m = k$	$d = O(\log(n/\varepsilon))$	extractor

Above, α is an arbitrarily small constant.

Figure 2: Comparison with best previous constructions

This compares with a degree bound of $(N/K) \cdot 2^{O((\log\log N)^9)}$ due to Ta-Shma [NT98]. Such expanders have applications to sorting and selecting in rounds, constructing superconcentrators, and constructing non-blocking networks [Pip87, AKSS89, WZ95], and the improvements of Corollary 3 translate to similar improvements in each of these applications. We remark that the construction of [WZ95] used to obtain Corollary 3 requires extractors that extract nearly all the entropy of the source.

The Trevisan extractor. The main tool in the Trevisan extractor is the Nisan–Wigderson generator [NW94], which builds a pseudorandom generator out of any Boolean function P such that the security of the pseudorandom generator is closely related to how hard P is to compute (on average). Let $\mathcal{S} = (S_1, \ldots, S_m)$ be a collection of subsets of $[d]$ each of size ℓ, and let $P\colon \{0,1\}^\ell \to \{0,1\}$ be any Boolean function. For a string $y \in \{0,1\}^d$, define $y|_{S_i}$ to be the string in $\{0,1\}^\ell$ obtained by projecting y onto the coordinates specified by S_i. Then the Nisan–Wigderson generator $\mathrm{NW}_{\mathcal{S},P}\colon \{0,1\}^d \to \{0,1\}^m$ is defined as

$$\mathrm{NW}_{\mathcal{S},P}(y) = P(y|_{S_1}) \cdots P(y|_{S_m}).$$

In the "indistinguishability proof" of [NW94], it is shown that for any function $D\colon \{0,1\}^m \to \{0,1\}$ which distinguishes the output of $\mathrm{NW}_{\mathcal{S},P}(y)$ (for uniformly selected y) from the uniform distribution on $\{0,1\}^m$, there is a "small" circuit C (or procedure of small "description size") such that $C^D(\cdot)$ (*i.e.* , C with oracle access to D) approximates $P(\cdot)$ reasonably well. It is shown that the size of the C is related to $\max_{i \neq j} |S_i \cap S_j|$, so one should use a collection of sets in which this quantity is small, while trying to minimize the seed length d.

We now give a rough description of the Trevisan extractor

$$\mathrm{EXT}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m.$$

For a string $u \in \{0,1\}^n$, let $\overline{u} \in \{0,1\}^{\overline{n}}$ be an encoding of u in an error-correcting code and define $\ell = \log \overline{n}$. We view $\overline{u}$ as a Boolean function $\overline{u}\colon \{0,1\}^\ell \to \{0,1\}$. As above, we fix a collection $\mathcal{S} = (S_1, \ldots, S_m)$ of subsets of $[d]$ of size ℓ.

Then the extractor is simply

$$\mathrm{EXT}_{\mathcal{S}}(u, y) = \mathrm{NW}_{\mathcal{S},\overline{u}}(y) = \overline{u}(y|_{S_1}) \cdots \overline{u}(y|_{S_m}).$$

The analysis of this extractor in [Tre98] shows that the output of this extractor is close to uniform as long as the source min-entropy required is greater than the size of the circuit built in the security reduction of [NW94]. Hence, one needs to make sure this circuit size is not much larger than the number m of output bits while minimizing the number d of truly random bits needed, which is equal to the seed length of the Nisan–Wigderson generator.

Our main improvements. The first improvement of this paper stems from the observation that actually $\max_i \sum_{j<i} 2^{|S_i \cap S_j|}$ is much better than $\max_{i \neq j} |S_i \cap S_j|$ as a measure of the size of the circuit built in the Nisan–Wigderson security reduction. So we are left with the problem of constructing set systems in which this quantity is small; we call such set systems *weak designs* (in contrast to *designs*, in which $\max_{i \neq j} |S_i \cap S_j|$ is bounded). We show that with weak designs, one can have d much smaller than is possible with the corresponding designs. The weak designs used in the first extractor of Theorem 1 are constructed using an application of the Probabilistic Method, which we then derandomize using the Method of Conditional Expectations (see [ASE92] and [MR95, Ch. 5]). We then apply a simple iteration to these first weak designs to obtain the weak designs used in the second extractor. We also prove a lower bound showing that our weak designs are near-optimal.

The second improvement is achieved by using a specific error-correcting code rather than an arbitrary one. More specifically, we use multilinear error-correcting codes over finite fields. In Trevisan's analysis for the size of the circuit C, the fact that $\overline{u}$ is an error-correcting code (rather than just an arbitrary function) is not used. The circuit complexity of the function $\overline{u}$, restricted to the subset of inputs $S_i \cap S_j$, is hence bounded by $\approx O(2^{|S_i \cap S_j|})$. Sometimes, however, this is a very bad upper bound. For example,

the circuit complexity of the function $\overline{u}$ itself (without restriction) is $\approx O(2^n)$ which is sometimes much smaller than $O(2^{\overline{n}})$. This gap is significant when ε is relatively small (because small ε requires an error-correcting code with very good distance properties, which in turn requires long codewords.) Here, we suggest that if one uses multilinear error-correcting codes and constructs the weak designs appropriately then the circuit complexity of the function $\overline{u}$, restricted to the subset of inputs $S_i \cap S_j$, can be bounded by a value much smaller than $2^{|S_i \cap S_j|}$.

Entropy loss. Since a (k,ε)-extractor $\text{EXT}: \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is given k bits of hidden randomness in its first input and d truly random bits in its second input, one can actually hope for the output length m to be almost $k+d$, rather than just k. The quantity $\Delta = k + d - m$ is therefore called the *entropy loss* of the extractor. Hence, in this language, the goal in constructing extractors is to simultaneously minimize both d and the entropy loss.

Nonconstructively, one can show that, for any n and $k \leq n$, there exist extractors $\text{EXT}_{n,k}: \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{k+d-\Delta}$ with $d = \log(n-k) + 2\log(1/\varepsilon) + O(1)$ and entropy loss $\Delta = 2\log(1/\varepsilon) + O(1)$, and these bounds on d and Δ are tight up to additive constants [RT97]. The explicit constructions, however, are still far from achieving these parameters. As for what is known, every entry in Figure 2 has an entropy loss of $k+d-m$, by definition. For example, the extractor of [GW97] has an entropy loss of $O(n-k+\log(1/\varepsilon))$ (which is only interesting when k is very close to n) and the extractor of [NT98] and the disperser of [TS98b] have entropy losses of polylog n. In addition, the "tiny families of hash functions" of [SZ98] give extractors with $d = O(k + \log n)$ and entropy loss $2\log(1/\varepsilon) + O(1)$; these have optimal entropy loss but are only interesting when k is very small (e.g., $k = \text{polylog}\, n$), as d is linear in k.

By combining the second extractors of Theorem 1 and Theorem 2 with extractors of [SZ98], we are able to achieve optimal entropy loss (up to an additive constant):

Theorem 4 *For every n, k, and ε such that $k \leq n$, there are explicit (k,ε)-extractors* $\text{EXT}: \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{k+d-\Delta}$ *with entropy loss*

$$\Delta = 2\log(1/\varepsilon) + O(1),$$

and

1. $d = O\left(\log^2(n/\varepsilon) \cdot \log k\right)$, *or*
2. $d = O\left(\log^2 n \cdot \log(1/\varepsilon) \cdot \log k\right)$.

In particular, in order for the output of the extractor to have statistical difference .01 from uniform, one need only lose a constant number of bits of entropy. A comparison of this result with previous results on entropy loss is given in Figure 3.

Improved pseudorandom generators. Using a relaxed notion of designs also gives some quantitative improvements over [NW94] in the construction of pseudorandom generators from hard Boolean functions. Details of these improvements are given in the full version of this paper.

2 Preliminaries

"log" indicates the logarithm base 2 and "ln" denotes the natural logarithm. If X is a probability distribution on a finite set, we write $x \leftarrow X$ to indicate that x is selected according to X. Two distributions X and Y on a set S are said to have *statistical difference* (or *variation distance*) ε if

$$\max_D |\Pr[D(X)=1] - \Pr[D(Y)=1]| = \varepsilon,$$

where the maximum is taken over all functions ("distinguishers") $D: \{0,1\}^m \to \{0,1\}$. A distribution X is said to have *min-entropy* k if for all x, $\Pr[X=x] \leq 2^{-k}$. It is useful to think of distributions of min-entropy k as being uniform over a subset of the domain of size 2^k.

We write U_j for the uniform distribution on strings of length j. A function $\text{EXT}: \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a *(k,ε)-extractor* if for every distribution X of min-entropy k, $\text{EXT}(X, U_d)$ has statistical difference at most ε from U_m. We say that a family of extractors $\{\text{EXT}_i: \{0,1\}^{n_i} \times \{0,1\}^{d_i} \to \{0,1\}^{m_i}\}_{i \in I}$ is *explicit* if EXT_i can be evaluated in time $\text{poly}(n_i, d_i)$.

3 Combinatorial designs

The combinatorial construction underlying the Nisan–Wigderson generator are combinatorial designs.

Definition 5 ([NW94]) [4] *A family of sets $S_1, \ldots, S_m \subset [d]$ is an (ℓ,ρ)-design if*

1. *For all i, $|S_i| = \ell$.*
2. *For all $i \neq j$, $|S_i \cap S_j| \leq \log \rho$.*

Motivation. In Trevisan's extractor, the parameters of a design correspond to the parameters of the extractor as follows (in the discussion below the parameter ε of the extractor is fixed, for simplicity, to be some small constant):

$$\begin{aligned} \text{source min-entropy} &\approx \rho m \\ \text{output length} &= m \\ \text{input length} &= 2^{\Theta(\ell)} \\ \text{additional randomness} &= d \end{aligned}$$

Hence, our goal in constructing designs is to minimize d given parameters m, ℓ, and ρ (such that $\rho > 1$). Notice that $1/\rho$ is essentially the fraction of the source min-entropy that is extracted, so ideally ρ would be as close to 1 as possible.

One explicit construction of designs is given by the following:

Lemma 6 ([NW94, Tre98]) *For every m, ℓ, and $\rho > 1$, there exists an efficiently constructible (ℓ,ρ)-design $S_1, \ldots, S_m \subset [d]$ with*

$$d = \frac{\ell^2 m^{O(1/\log \rho)}}{\log \rho}.$$

Notice that the dependence on ρ is very poor. In particular, if we want to extract a constant fraction of the min-entropy, we need more than m^c truly random bits for some $c > 0$. This is unavoidable with the current definition of designs: if $\rho < 2$, then all the sets must be disjoint, so $d \geq m\ell$. In general, we have the following lower bound, proved in the full version of the paper.

Proposition 7 *If $S_1, \ldots, S_m \subset [d]$ is an (ℓ,ρ)-design, then*

$$d \geq m^{1/\log 2\rho} \cdot (\ell - \log \rho)$$

The first improvement of this paper stems from the observation that actually a weaker form of design suffices for the Nisan–Wigderson generator and the construction of extractors:

[4] There is a somewhat related notion in the combinatorics literature known as a *2-design* (see, e.g. [AK92]). In 2-designs, strong additional regularity requirements are imposed (such as all the pairwise intersections being *exactly* the same size and all points being contained in the same number of sets). These additional requirements are irrelevant in our applications.

reference	additional randomness d	entropy loss Δ	type
[GW97]	$d = O(n - k + \log(1/\varepsilon))$	$\Delta = n - k + 12\log(1/\varepsilon) + O(1)$	extractor
[SZ98]	$d = O(k + \log n)$	$\Delta = 2\log(1/\varepsilon) + O(1)$	extractor
[NT98]	$d = O(\log^9 n \cdot \log(1/\varepsilon))$	$\Delta = O(\log^9 n \cdot \log(1/\varepsilon))$	extractor
[TS98b]	$d = O(\log(n/\varepsilon))$	$\Delta = \text{polylog}(n/\varepsilon)$	disperser
this paper	$d = O((\log^2(n/\varepsilon))(\log k))$	$\Delta = 2\log(1/\varepsilon) + O(1)$	extractor
nonconstructive & optimal [RT97]	$d = \log(n - k) + O(1)$	$\Delta = 2\log(1/\varepsilon) + O(1)$	extractor

All of the above work for any source of min-entropy k.

Figure 3: Results on entropy loss

Definition 8 *A family of sets $S_1, \ldots, S_m \subset [d]$ is a* weak (ℓ, ρ)-design *if*

1. *For all i, $|S_i| = \ell$.*
2. *For all i,*
$$\sum_{j<i} 2^{|S_i \cap S_j|} \leq \rho \cdot (m-1).$$

We will show that the parameters of a weak design correspond to the parameters of our extractors in the same way that designs corresponded to the parameters of Trevisan's extractor. Notice that every (ℓ, ρ)-design is a weak (ℓ, ρ)-design. But one can, for many settings of m, ℓ, and ρ, achieve weak (ℓ, ρ)-designs $S_1, \ldots, S_m \subset [d]$ with much smaller values of d than possible with (ℓ, ρ)-designs. Indeed, we will prove the following in Section 6 using a probabilistic argument:

Lemma 9 *For every $\ell, m \in \mathbb{N}$ and $\rho > 1$, there exists a weak (ℓ, ρ)-design $S_1, \ldots, S_m \subset [d]$ with*

$$d = \left\lceil \frac{\ell}{\ln \rho} \right\rceil \cdot \ell.$$

Moreover, such a family can be found in time $\text{poly}(m, d)$.

This is already much better than what is given by Lemma 6; for constant ρ, d is $O(\ell^2)$ instead of $\ell^2 m^c$. However, as ρ gets very close to 1, d gets very large. Specifically, if $\rho = 1 + \gamma$ for small γ, then the above gives $d = O(\ell^2/\gamma)$. To improve this, we notice that the proof of Lemma 9 does not take advantage of the fact that there are fewer terms in $\sum_{j<i} 2^{|S_i \cap S_j|}$ when i is small; indeed the proof actually shows how to obtain $\sum_{j<i} 2^{|S_i \cap S_j|} < \rho \cdot (i-1)$. Since we only need a bound of $\rho \cdot (m-1)$ for all i, this suggests that we should "pack" more sets in the beginning. This packing is accomplished by iterating the construction of Lemma 9 (directly inspired by the iteration of Wigderson and Zuckerman [WZ95] on extractors), and yields the following improvement.

Lemma 10 *For every $\ell, m \in \mathbb{N}$ and $3/m \leq \gamma < 1/2$, there exists a weak $(m, \ell, 1+\gamma, d)$-design with*

$$d = O\left(\ell^2 \log \frac{1}{\gamma}\right).$$

Moreover, such a family can be found in time $\text{poly}(m, d)$.

In particular, we can take $\gamma = \Theta(1/m)$ and extract essentially all of the entropy of the source using $d = O(\ell^2 \log m)$ truly random bits. Lemma 10 will be proven in Section 6.

For extractors which use only $O(\log n)$ truly random bits, where n is the input length, one would need $d = O(\ell)$. However, one cannot hope to do better than $\Omega(\ell^2)$ using the current analysis with weak designs. Indeed, the following proposition, proved in the full version of the paper, shows that our weak designs are optimal up to the $\log(1/\gamma)$ factor in our second construction.

Proposition 11 *For every (ℓ, ρ)-weak design $S_1, \ldots, S_m \subset [d]$,*

$$d \geq \min\left(\frac{\ell^2}{2\log 2\rho}, \frac{m\ell}{2}\right)$$

Notice that $d = m\ell$ can be trivially achieved having all the sets disjoint and that $\log 2\rho$ approaches 1 as ρ approaches 1, so the lower bound for $\rho \approx 1$ is essentially $\Omega(\ell^2)$.

4 The extractor

In this section, we describe the Trevisan extractor and analyze its performance when used with our weak designs. The description of the extractor follows [Tre98] very closely. The main tool in the Trevisan extractor is the Nisan–Wigderson generator [NW94]. Let $S = (S_1, \ldots, S_m)$ be a collection of subsets of $[d]$ of size ℓ, and let $P: \{0,1\}^\ell \to \{0,1\}$ be any Boolean function. For a string $y \in \{0,1\}^d$, define $y|_{S_i}$ to be the string in $\{0,1\}^\ell$ obtained by projecting y onto the coordinates specified by S_i. Then the Nisan–Wigderson generator $\text{NW}_{S,P}$ is defined as

$$\text{NW}_{S,P}(y) = P(y|_{S_1}) \cdots P(y|_{S_m}).$$

In addition to the Nisan–Wigderson generator, the Trevisan extractor makes use of error-correcting codes:

Lemma 12 (error-correcting codes) *For every n and δ there is a code $\text{EC}_{n,\delta}: \{0,1\}^n \to \{0,1\}^{\overline{n}}$ where $\overline{n} = \text{poly}(n, 1/\delta)$ such that every Hamming ball of relative radius $1/2 - \delta$ in $\{0,1\}^{\overline{n}}$ contains at most $1/\delta^2$ codewords. Furthermore, $\text{EC}_{n,\delta}$ can be evaluated in time $\text{poly}(n, 1/\delta)$ and $\overline{n}$ can be assumed to be a power of 2.*

We can now describe the Trevisan extractor, which takes as parameters n, m, k, and ε, where $m \leq k \leq n$. Let $\text{EC}: \{0,1\}^n \to \{0,1\}^{\overline{n}}$ be as in Lemma 12, with $\delta = \varepsilon/4m$ and define $\ell = \log \overline{n} = O(\log n/\varepsilon)$. For $u \in \{0,1\}^n$, we view $\text{EC}(u)$ as a Boolean function $\overline{u}: \{0,1\}^\ell \to \{0,1\}$. Let $\mathcal{S} = (S_1, \ldots, S_m)$ be a collection of subsets of $[d]$ (for some d) such that $|S_i| = \ell$ for each i. (How $\mathcal{S}$ is selected will crucially affect the performance of the extractor; we will later choose it to be one of our weak designs.)

Then the extractor $\text{EXT}_{\mathcal{S}}: \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is defined as

$$\text{EXT}_{\mathcal{S}}(u,y) = \text{NW}_{\mathcal{S},\overline{u}}(y) = \overline{u}(y|_{S_1}) \cdots \overline{u}(y|_{S_m}).$$

We will now analyze this extractor. The following lemma, due to Yao, allows us to focus on "next-bit predictors" instead of distinguishers.

Lemma 13 ([Yao82]) *Suppose that Z is a distribution on $\{0,1\}^m$ whose statistical difference from U_m is greater than ε. Then there is an $i \in [m]$ and a function ("next-bit predicator") $A: \{0,1\}^{i-1} \to \{0,1\}$ such that*

$$\Pr_{z \leftarrow Z}[A(z_1 z_2 \cdots z_{i-1}) = z_i] > \frac{1}{2} + \frac{\varepsilon}{m}.$$

The following lemma is a refinement of ones in [NW94, Tre98]. It shows how, from any next-bit predictor A for $\text{NW}_{\mathcal{S},P}$, one can obtain a "program" of small description size (or circuit complexity) which, using A as an oracle, computes P with noticeable advantage.

Lemma 14 *Fix $\mathcal{S}$. For every $i \in [m]$, there is a set $\mathcal{F}_i$ of functions from $\{0,1\}^\ell$ to $\{0,1\}^{i-1}$ (depending only on $\mathcal{S}$ and i) such that*

1. *For every function $P: \{0,1\}^\ell \to \{0,1\}$ and every predictor $A: \{0,1\}^{i-1} \to \{0,1\}$, there exists a function $f \in \mathcal{F}_i$ such that*

$$\begin{aligned}&\Pr_x[A(f(x)) = P(x)] \\ &\quad \geq \Pr_y\left[A\left(P(y|_{S_1}) \cdots P(y|_{S_{i-1}})\right) = P(y|_{S_i})\right],\end{aligned}$$

   where x is selected uniformly from $\{0,1\}^\ell$ and y from $\{0,1\}^d$.

2. $\log|\mathcal{F}_i| \leq \sum_{j<i} 2^{|S_i \cap S_j|}$.

The improvement over [NW94, Tre98] in Lemma 14 is the use of $\sum_{j<i} 2^{|S_i \cap S_j|}$ rather than $m \cdot 2^{\max_j |S_i \cap S_j|}$ in the bound on $|\mathcal{F}_i|$. This refined bound illustrates the connection with weak designs.

Proof: Let

$$\alpha = \Pr_y\left[A\left(P(y|_{S_1}) \cdots P(y|_{S_{i-1}})\right) = P(y|_{S_i})\right]$$

By an averaging argument we can fix all the bits of y outside S_i while preserving the prediction probability. Renaming $y|_{S_i}$ as x, we now observe that x varies uniformly over $\{0,1\}^\ell$ while $P(y|_{S_j})$ for $j \neq i$ is now a function P_j of x that depends on only $|S_i \cap S_j|$ bits of x. So, we have

$$\Pr_x[A(P_1(x) \cdots P_{i-1}(x)) = P(x)] \geq \alpha.$$

Therefore, it suffices to let $\mathcal{F}_i$ be the set of functions f of the form $x \mapsto (P_1(x), P_2(x), \ldots, P_{i-1}(x))$, where $P_j(x)$ depends only some set T_{ij} of bits of x, where $|T_{ij}| = |S_i \cap S_j|$. The number of bits it takes to represent each P_j is $2^{|T_{ij}|} = 2^{|S_i \cap S_j|}$. So, the total number of bits it takes to represent a function in $\mathcal{F}_i$ is at most $\sum_{j<i} 2^{|S_i \cap S_j|}$, giving the desired bound on $\log|\mathcal{F}_i|$. ■

We now analyze the extractor $\text{EXT}_{\mathcal{S}}$ when we take $\mathcal{S}$ to be a weak design.

Proposition 15 *If $\mathcal{S} = (S_1, \ldots, S_m)$ (with $S_i \subset [d]$) is a weak (ℓ, ρ)-design for $\rho = (k - 3\log(m/\varepsilon) - 5)/m$, then $\text{EXT}_{\mathcal{S}}: \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ε)-extractor.*

The proof of Proposition 15 basically follows the analysis of Trevisan's extractor in [Tre98] except that we use the more refined bounds on $|\mathcal{F}_i|$ given by Lemma 14.

Proof: *Let X be any distribution of min-entropy k. We need to* show that the statistical difference between U_m and $\text{EXT}(X, U_d)$ is at most ε. By Lemma 13, it suffices to show that for every next-bit predictor $A: \{0,1\}^{i-1} \to \{0,1\}$,

$$\Pr_{u \leftarrow X, y}\left[A\left(\overline{u}(y|_{S_1}) \cdots \overline{u}(y|_{S_{i-1}})\right) = \overline{u}(y|_{S_i})\right] \leq \frac{1}{2} + \frac{\varepsilon}{m}$$

where y is selected uniformly from $\{0,1\}^d$. So let $A: \{0,1\}^{i-1} \to \{0,1\}$ be any next-bit predictor and let $\mathcal{F}_i$ be as in Lemma 14, so that $|\mathcal{F}_i| \leq 2^{\rho m}$.

Let B be the set of u for which there exists an $f \in \mathcal{F}_i$ such that $\Pr_x[A(f(x)) = \overline{u}(x)] > 1/2 + \varepsilon/2m$. In other words, B is the set of "bad" u for which $\overline{u}$ can be easily approximated given oracle access to A. By the property of the error-correcting code given in Lemma 12, for each function $f \in \mathcal{F}_i$, there are at most $(4m/\varepsilon)^2$ strings $u \in \{0,1\}^n$ such that $\Pr_x[A(f(x)) = \overline{u}(x)] > 1/2 + \varepsilon/2m$. By the union bound,

$$|B| \leq (4m/\varepsilon)^2 \cdot |\mathcal{F}_i| \leq (4m/\varepsilon)^2 \cdot 2^{\rho m}.$$

Since X has min-entropy k, each $u \in B$ has probability at most 2^{-k} of being selected from X, so

$$\begin{aligned}\Pr_{u \leftarrow X}[u \in B] &\leq \left((4m/\varepsilon)^2 2^{\rho m}\right) \cdot 2^{-k} \\ &= \left((4m/\varepsilon)^2 2^{k - 3\log(m/\varepsilon) - 5}\right) \cdot 2^{-k} \\ &= \varepsilon/2m\end{aligned}$$

Now, by Lemma 14, if $u \notin B$, then

$$\Pr_y\left[A\left(\overline{u}(y|_{S_1}) \cdots \overline{u}(y|_{S_{i-1}})\right) = \overline{u}(y|_{S_i})\right] \leq \frac{1}{2} + \frac{\varepsilon}{2m}.$$

Thus,

$$\begin{aligned}&\Pr_{u \leftarrow X, y}\left[A\left(\overline{u}(y|_{S_1}) \cdots \overline{u}(y|_{S_{i-1}})\right) = \overline{u}(y|_{S_i})\right] \\ &\leq \Pr_{u \leftarrow X}[u \in B] + \Pr_{u \leftarrow X}[u \notin B]\left(\frac{1}{2} + \frac{\varepsilon}{2m}\right) \\ &\leq \frac{\varepsilon}{2m} + \left(\frac{1}{2} + \frac{\varepsilon}{2m}\right) \\ &= \frac{1}{2} + \frac{\varepsilon}{m}.\end{aligned}$$

■

Combining Proposition 15 with the weak designs given by Lemmas 9 and 10 essentially proves Theorem 1. The only technicality is that Proposition 15 does not allow us to take $\rho = k/m$ (or $k/(m-1)$) which is what we would need to deduce Theorem 1 directly. Instead, we use $\rho = (k - \Delta)/m$ and consequently lose $\Delta = 3\log(m/\varepsilon) + 5$ bits of the source entropy in Proposition 15. However, since Δ is so small, we can give our extractor Δ more truly random bits in its seed (increasing d by only a $o(d)$ additive term) which we just concatenate to the output to compensate for the loss. The details of this are given below.

Proof of Theorem 1: Let $\Delta = 3\log(m/\varepsilon) + 5$. Let $k' = k - \Delta$, $m' = m - \Delta - 3$, and $\rho = k'/m' > k/(m-1)$. For Part 1 (resp., Part 2), apply Proposition 15 with the weak (ℓ, ρ)-design $S_1, \ldots, S_{m'} \subset [d']$ of Lemma 9 (resp., Lemma 10). This

gives an (k, ε)-extractor EXT: $\{0,1\}^n \times \{0,1\}^{d'} \to \{0,1\}^{m'}$, with $d' = O\left(\frac{\log^2(n/\varepsilon)}{\log(k/m)}\right)$ (resp., $d' = O(\log^2(n/\varepsilon)\log(1/\gamma))$). By using $\Delta + 3$ additional bits in the seed and simply concatenating these to the output, we obtain a (k, ε)-extractor EXT: $\{0,1\}^n \times \{0,1\}^{d'+\Delta+3} \to \{0,1\}^m$, as desired. (In applying Lemma 10, we need to make sure that $\rho < 3/2$, but if $\rho \geq 3/2$, we can use the weak design of Lemma 9 instead.) ■

5 Reducing the error

The construction given above works well and improves over previous constructions when ε is relatively large. However, the number d of truly random bits needed is quadratic in $\log(1/\varepsilon)$, which is not as good as the linear dependency achieved by some previous constructions. In this section, we improve this quadratic dependency in our constructions (and in Trevisan's construction) to a linear dependency. We only sketch the proof in this section, due to space constraints and the fact that the results have been superseded by our recent work (in preparation).

The quadratic dependence on $\log(1/\varepsilon)$ in our extractor arises from the fact that an (ℓ, ρ)-weak design requires a universe whose size grows quadratically with ℓ (cf., Proposition 11). In the extractor of the previous section (and Trevisan's extractor), ℓ is taken to be the logarithm of the length of the error-correcting code used (as we view codewords as functions P: $\{0,1\}^\ell \to \{0,1\}$). The analysis of the extractor reveals that in order to achieve a small statistical difference ε from uniform, we must use an error-correcting code with very good distance properties; namely, one in which no Hamming ball of radius $1/2 - O(\varepsilon/m)$ contains many codewords. However, an error-correcting code with such a strong distance property must have length at least $\text{poly}(n, \varepsilon)$, resulting in $\ell = \Omega(\log(n/\varepsilon))$, and a seed length that is quadratic in $\log(1/\varepsilon)$.

The solution we give in this section is to use an error-correcting code over a large alphabet F, in which we view every codeword as a function from F^ℓ to F rather than a function from $\{0,1\}^\ell$ to $\{0,1\}$. Then it is possible to have a code with very good distance properties (relative to ε) with ℓ being independent of ε; only the alphabet size F need depend on ε. Using this approach, we encounter two problems. The first problem is that the function which computes the codeword P given a predictor A (as in Lemma 14) will be built from functions of the form P_j: $F^{|S_i \cap S_j|} \to F$. In the proof of Lemma 14, we bounded the description size of the P_j's by the description size of an arbitrary function $F^{|S_i \cap S_j|} \to F$, which is $2^{|S_i \cap S_j|}$ when $F = \{0,1\}$. But, as F increases in size, this bound on description size becomes too large to handle. The second problem is that, when we use a large alphabet, the output of the extractor consists of elements of F rather than bits. We will not be able to argue that these elements of F are uniformly distributed, but rather that the i'th element of F in the output is unpredictable given the first $i-1$ elements of F.

The solution to the first problem comes from our choice of error-correcting codes. We use multilinear error correcting codes (over finite fields) rather than the arbitrary error correcting codes used in Section 4. We can then make use of the fact that the restriction of a multilinear function to a subset of its input variables is still a multilinear function. We can hence bound the description size of that restriction by the description size of a multilinear function rather than the description size of an arbitrary function.

The second problem can be solved using standard techniques. Specifically, the fact that the i'th component of the output is unpredictable given the first $i-1$ components means that the output is what is known as a *block-wise source* [CG88]. In our case, the block-wise source has blocks of logarithmic length, and standard techniques can be used to extract truly random bits from such a source using a small number of additional truly random bits.

Let F be some fixed finite field such that $\log|F| \approx c \cdot \log(n/\varepsilon)$, where c is some sufficiently large constant (say $c = 10$). For $\varepsilon \geq 1/n$, the dependence on ε in the extractors of Theorem 1 can be absorbed into the hidden constant. Thus, we will only need to use the constructions of this section in case $\varepsilon < 1/n$, and hence we may assume that

$$\log|F| = O(\log(1/\varepsilon)).$$

In this section, we think of an extractor EXT: $\{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ as a function

$$\text{EXT}: F^{n'} \times F^{d'} \to F^{m'},$$

where $n' = n/(\log|F|)$, $d' = d/(\log|F|)$ and $m' = m/(\log|F|)$ (we assume for simplicity that $n', d', m', \log n'$, and $\log|F|$ are all integer).

Let $\mathcal{S} = (S_1, \ldots, S_{m'})$ be a collection of subsets of $[d']$ such that $|S_i| = \ell$ for each i, and let P: $F^\ell \to F$ be any function. For a string $y \in F^{d'}$, define $y|_{S_i}$ to be the string in F^ℓ obtained by projecting y onto the coordinates specified by S_i. Then we define $\text{NW}'_{\mathcal{S},P}$ as

$$\text{NW}'_{\mathcal{S},P}(y) = P(y|_{S_1}) \cdots P(y|_{S_{m'}}).$$

We will use in this section $\ell = \log n'$; note that ℓ is bounded by $\log n$, independent of ε. Let G be the set of all functions from F^ℓ to F. There are $|F|^{2^\ell} = |F|^{n'}$ *multilinear* functions from F^ℓ to F (one needs to specify 2^ℓ coefficients), so we may define an error-correcting code EC: $F^{n'} \to G$ which associates to each element u of $F^{n'}$ a distinct multilinear function $\text{EC}(u) = \overline{u}$: $F^\ell \to F$. The distance property of this code is formalized by the following standard fact:

Lemma 16 *For every function Q: $F^\ell \to F$, there are at most $O(\sqrt{|F|/\ell})$ codewords (i.e., multilinear functions) that agree with Q in at least a $\sqrt{2\ell/|F|}$ fraction of the points in F^ℓ.*

We define the function $\text{EXT}_{\mathcal{S}}$: $F^{n'} \times F^{d'} \to F^{m'}$ as

$$\text{EXT}_{\mathcal{S}}(u, y) = \text{NW}'_{\mathcal{S},\overline{u}}(y) = \overline{u}(y|_{S_1}) \cdots \overline{u}(y|_{S_{m'}}).$$

(The function EXT is still not our final extractor). Note that the number of truly random bits used by EXT is $d' \log|F| = O(d' \cdot \log(1/\varepsilon))$. The following lemma is analogous to Lemma 14. It shows how, from any next-element predictor A for $\text{NW}'_{\mathcal{S},P}$, one can obtain a "program" of small description size (or circuit complexity) which, using A as an oracle, computes P with noticeable advantage.

Lemma 17 *Fix $\mathcal{S}$. For every $i \in [m']$, there is a set $\mathcal{F}_i$ of functions from F^ℓ to F^{i-1} (depending only on $F, \mathcal{S}$ and i) such that*

1. *For every multilinear function P: $F^\ell \to F$ and every predictor A: $F^{i-1} \to F$, there exists a function $f \in \mathcal{F}_i$ such that*

$$\Pr_x[A(f(x)) = P(x)] \geq \Pr_y\left[A\left(P(y|_{S_1}) \cdots P(y|_{S_{i-1}})\right) = P(y|_{S_i})\right],$$

where x is selected uniformly from F^ℓ and y from $F^{d'}$.

2. $\log|\mathcal{F}_i| \le \sum_{j<i} 2^{|S_i \cap S_j|} \cdot \log|F|$.

For the proof, we use the fact that the restriction of a multilinear function to a subset of its input variables is a multilinear function, and the fact that the logarithm of the number of multilinear functions in $|S_i \cap S_j|$ variables is $2^{|S_i \cap S_j|} \cdot \log|F|$. Otherwise, the proof is similar to the one of Lemma 14.

Now assume that $\mathcal{S}$ is a weak (ℓ, ρ)-design for $\rho = (k - c \cdot \log|F|)/m$ (where, say, $c = 10$), and let X be any distribution of min-entropy k. The following proposition shows that $\text{EXT}(X, U_d)$ doesn't have a good next-element predictor. The proposition is analogous to Proposition 15.

Proposition 18 *If $\mathcal{S} = (S_1, \ldots, S_{m'})$ (with $S_i \subset [d']$) is a weak (ℓ, ρ)-design for $\rho = (k - c \cdot \log|F|)/m$ (where c is some sufficiently large constant, say $c = 10$), and X is a distribution of min-entropy k then for every next-element predictor $A: F^{i-1} \to F$,*

$$\Pr_{u \leftarrow X, y}\left[A\left(\overline{u}(y|_{S_1}) \cdots \overline{u}(y|_{S_{i-1}})\right) = \overline{u}(y|_{S_i})\right] \le \frac{1}{|F|^{\delta}}$$

where δ is some (not too small) constant (say $\delta = 1/4$), and where y is selected uniformly from $F^{d'}$.

The proof is similar to the one of Proposition 15, except that we use the distance property of multilinear error-correcting codes given by Lemma 16 and we use Lemma 17 rather than Lemma 14.

In general, the function $\text{EXT}_{\mathcal{S}}$ is not a good extractor. Nevertheless, by Proposition 18, we know that each element $\text{EXT}(X, U_d)$ has large min-entropy ($\delta \log|F|$ bits) given all its predecessors. That is, it is a "block-wise source" in the sense of [CG88], in which the min-entropy of each block given the predecessors is a constant fraction of its length (which is $\log|F|$). We can now construct an extractor from $\text{EXT}_{\mathcal{S}}$ in one of the following ways:

1. By applying on the entire output $\text{EXT}(X, U_d)$ the extractor of [Zuc97] that extracts a constant fraction of the min-entropy as long as the min-entropy is at least a constant fraction of the number of bits.
2. By applying on each element of $\text{EXT}(X, U_d)$ a pairwise independent hash function $h : \{0,1\}^{\log|F|} \to \{0,1\}^{\delta' \cdot \log|F|}$, where δ' is some small constant (we can apply the same hash function on all the elements).

Both ways are very efficient in terms of the number of additional random bits needed.

The first part of Theorem 2 is now obtained by using the weak designs given by Lemmas 9 (as in the proof of Theorem 1). The resulting seed length (using an (ℓ, ρ)-weak design for $\rho = (k - c \log|F|)/m$) is

$$d = O(d' \log(1/\varepsilon)) = O\left(\frac{\ell^2}{\log \rho} \cdot \log(1/\varepsilon)\right).$$

However, the number of bits we extract is only $\delta' \cdot \log|F| \cdot m' = \delta' m \approx \delta' k/\rho$, for some constant $\delta' < 1$. Hence, we can only directly use this to extract upto a small constant fraction of the min-entropy (even if we use the weak designs of Lemma 10). In order to extract more of the min-entropy of the source, we will need to use iterations, as in [WZ95]. A constant number of iterations will allow us to extract any constant fraction of the min-entropy. In general, to obtain $m = k/(1+\gamma)$, we will need $O(\log(1/\gamma))$ iterations and hence we need $O\left(\log^2 n \cdot \log(1/\varepsilon) \cdot \log(1/\gamma)\right)$ additional random bits.

6 Construction of weak designs

Proof of Lemma 9: Let ℓ, m, and ρ be given, and let $d = \lceil \ell / \ln \rho \rceil \cdot \ell$. We view $[d]$ as the disjoint union of ℓ blocks $B_1, \ldots, B_\ell$ each of size $\lceil \ell / \ln \rho \rceil$. We construct the sets $S_1, \ldots, S_m$ in sequence so that

1. Each set contains exactly one element from each block, and
2. $\sum_{j<i} 2^{|S_i \cap S_j|} \le \rho \cdot (i-1)$.

Supppose we have $S_1, \ldots, S_{i-1} \subset [d]$ satisfying the above conditions. We prove that there exists a set S_i satisfying the required conditions using the Probabilistic Method [ASE92] (see also [MR95, Ch. 5]). Let $a_1, \ldots, a_\ell$ be uniformly and independently selected elements of $B_1, \ldots, B_\ell$, respectively, and then let $S_i = \{a_1, \ldots, a_\ell\}$. We will argue that with nonzero probability, Condition 2 holds. Let $Y_{j,k}$ be the indicator random variable for the event $a_k \in S_j$, so $\Pr[Y_{j,k} = 1] = 1/|B_k| = 1/\lceil \ell / \ln \rho \rceil$. Notice that for a fixed j, the random variables $Y_{j,1}, \ldots, Y_{j,\ell}$ are independent. A straightforward calculation shows that

$$\mathrm{E}\left[\sum_{j<i} 2^{|S_i \cap S_j|}\right] = \sum_{j<i} \prod_{k} \mathrm{E}\left[2^{Y_{j,k}}\right] \le (i-1) \cdot \rho.$$

Hence, with nonzero probability, Condition 2 holds, so a set S_i satisfying the requirements exists. However, we want to find such a set deterministically. This can be accomplished by a straightforward application of the Method of Conditional Expectations (see [ASE92] and [MR95, Ch. 5]). Details can be found in the full version of the paper. ■

Proof of Lemma 10: For simplicity, assume that $1 + \gamma = 1/(1 - 2^{-h})$ and $m = 2^q/(1+\gamma)$. Let $d_0 = \lceil \ell / \ln 2 \rceil \cdot \ell$ and let $d = h \cdot d_0 = O(\ell^2 \cdot \log(1+\gamma))$. We view $[d]$ as the disjoint union of h blocks $B_1, \ldots, B_h$ each of size d_0. For each $t \in [h]$, let $m_t = 2^{q-t}$ and $n_t = \sum_{s=1}^{t-1} m_s$, so $\sum_t m_t = m$.

Now we define our weak design $S_1, \ldots, S_m$. For each $t \in [h]$, we let $S_{n_t+1}, \ldots, S_{n_t+m_t} \subset B_t$ be a weak $(\ell, 2)$-design as given by Lemma 9. In other words, we take the ordered union of h weak $(\ell, 2)$-designs (consisting of $m_1, m_2, \ldots, m_h$ sets, respectively) using disjoint subsets of the universe for each. The number of sets is m, the size of the universe is d, and each set is of size ℓ, so we only need to check that for all $i \in [m]$, $\sum_{j<i} 2^{|S_i \cap S_j|} < \rho \cdot (m-1)$. For $i \in \{n_t + 1, \ldots, n_t + m_t\}$, S_i is disjoint from any S_j for any $j \le n_t$ and

$$\sum_{j=n_t+1}^{i-1} 2^{|S_i \cap S_j|} \le 2 \cdot (m_t - 1)$$

since $S_{n_t+1}, \ldots, S_{n_t+m_t}$ is a weak $(\ell, 2)$-design. Thus, we have

$$\begin{aligned}\sum_{j<i} 2^{|S_i \cap S_j|} &= \sum_{j=1}^{n_t} 2^{|S_i \cap S_j|} + \sum_{j=n_t+1}^{i-1} 2^{|S_i \cap S_j|} \\ &\le n_t + 2 \cdot (m_t - 1) \\ &= 2^q - 2 < (1+\gamma)(m-1),\end{aligned}$$

as desired. ■

7 Achieving optimal entropy loss

Recall that the *entropy loss* of an extractor $\text{EXT}: \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is defined as $\Delta = k+d-m$, and we can hope for this to be as small as $2\log(1/\varepsilon)+O(1)$ with $d = \log(n-k)+O(1)$ [RT97].

In constructing our extractor $\text{EXT}_S(u,y) = \text{NW}_{S,\overline{u}}(y)$, we "threw away" y after using it as a seed for the Nisan–Wigderson generator and hence the d bits of entropy carried by y were lost. However, the analysis of the Nisan–Wigderson generator actually shows that the quality of the generator is not affected if the seed is revealed. Thus, we define $\text{EXT}'_S(u,y) = (y, \text{NW}_{S,\overline{u}}(y))$. Now all the analysis of EXT done in Section 4 actually applies to EXT′ (in Lemma 14, give the predictor A the seed y in addition to $\text{NW}_{S,P}(y)$), and we obtain the following strengthening of Proposition 15:

Proposition 19 *If $\mathcal{S} = (S_1, \ldots, S_m)$ (with $S_i \subset [d]$) is a weak (ℓ,ρ)-design for $\rho = (k-3\log(m/\varepsilon)-5)/m$, then $\text{EXT}'_{\mathcal{S}}: \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{m+d}$ is a (k,ε)-extractor.*

Combining Proposition 19 and Lemma 10 with $m = k - 3\log(k/\varepsilon) - 6$ immediately gives us logarithmic entropy loss:

Proposition 20 *For every n, k, and ε such that $k \le n$, there is an explicit (k,ε)-extractor* $\text{EXT}: \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{k+d-\Delta}$ *with*

$$d = O\left(\left(\log^2(n/\varepsilon)\right)(\log k)\right)$$

and entropy loss

$$\Delta = 3\log(k/\varepsilon) + O(1)$$

By doing a finer analysis than the one done in Section 4, the entropy loss can be reduced to $2\log(k/\varepsilon) + O(1)$, but we will use a different method to make the entropy loss even better. We use a slight modification of an idea due to Wigderson and Zuckerman [WZ95]: Suppose we have a (k,ε)-extractor $\text{EXT}: \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{k+d-\Delta}$ with entropy loss Δ. Now, if x is taken from a source of min-entropy k and y from the uniform distribution on $\{0,1\}^d$, then conditioned on "most" values of $\text{EXT}(X, U_d)$, the pair (x,y) will still have min-entropy close to Δ. So, we can use a different extractor (with fresh truly random bits) to extract some more of this min-entropy. This is formalized by the following lemma (proved in the full version of the paper), which strengthens the one in [WZ95]:

Lemma 21 *Let $s > 0$. Suppose*

$$\text{EXT}_1: \{0,1\}^n \times \{0,1\}^{d_1} \to \{0,1\}^{m_1}$$

is a (k,ε_1)-extractor with entropy loss Δ_1 and

$$\text{EXT}_2: \{0,1\}^{n+d_1} \times \{0,1\}^{d_2} \to \{0,1\}^{m_2}$$

is a $(\Delta_1 - s, \varepsilon_2)$-extractor with entropy loss Δ_2. Define

$$\text{EXT}: \{0,1\}^n \times \{0,1\}^{d_1+d_2} \to \{0,1\}^{m_1+m_2}$$

by

$$\text{EXT}(x,(y_1,y_2)) = \text{EXT}_1(x,y_1) \circ \text{EXT}_2((x,y_1),y_2),$$

where $\circ$ denotes concatenation. Then EXT *is a*

$$\left(k, \left(\tfrac{1}{1-2^{-s}}\right)\cdot\varepsilon_1 + \varepsilon_2\right)\text{-extractor}$$

with entropy loss $\Delta_2 + s$.

Lemma 21 has two main differences from the one in [WZ95]: First, we use the second extractor on the pair (x, y_1) rather than just x; this enables us to make the output length close to $k + d_1 + d_2$ rather than just k. Second, the statistical difference from uniform in EXT has a better dependence on s (in [WZ95], the expression is $\varepsilon_1 + \varepsilon_2 + 2^{-s}$). Now let us see how Lemma 21 can be used to make our entropy loss optimal. If we use the extractor given by Proposition 20 as EXT_1, Lemma 21 tells us that we need only find an extractor EXT_2 which works well for very small (i.e., logarithmic) min-entropy. The following extractor of Srinivasan and Zuckerman [SZ98] achieves exactly what we want:

Lemma 22 ([SZ98]) *For every n, $k \le n$, and $\varepsilon > 0$, there is an explicit (k,ε)-extractor* $\text{EXT}: \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{k+d-\Delta}$ *with entropy loss $\Delta = 2\lceil\log(1/\varepsilon)\rceil + 2$ and $d = O(k + \log n)$.*

The first extractor of Theorem 4 follows from combining the extractors of Proposition 20 and Lemma 22 via Lemma 21. The second extractor is obtained in a similar manner, combining the extractors of Theorem 2 with those of Lemma 22 instead. Details are given in the full version of the paper.

Acknowledgments

The third author is most grateful to Luca Trevisan for sharing his novel insights into these problems with him. He also thanks Oded Goldreich for his neverending supply of guidance and support, and for all his help with the presentation. While doing this work, the third author had many useful discussions with Luca, Oded, and several other people — including Shafi Goldwasser, Adam Klivans, Madhu Sudan, Amnon Ta-Shma, Avi Wigderson, and David Zuckerman. Some of these discussions led to specific improvements in this paper, which are not enumerated for the sake of brevity.

References

[ACR97] Alexander E. Andreev, Andrea E. F. Clementi, and José D. P. Rolim. Worst-case hardness suffices for derandomization: A new method for hardness-randomness trade-offs. In Pierpaolo Degano, Robert Gorrieri, and Alberto Marchetti-Spaccamela, editors, *Automata, Languages and Programming, 24th International Colloquium*, volume 1256 of *Lecture Notes in Computer Science*, pages 177–187, Bologna, Italy, 7–11 July 1997. Springer-Verlag.

[AK92] E.F. Assmus and J.D. Key. *Designs and their codes.* Number 103 in Cambridge Tracts in Mathematics. Cambridge University Press, 1992.

[AKSS89] Miklós Ajtai, János Komlós, William Steiger, and Endre Szemerédi. Almost sorting in one round. In Silvio Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 117–125. JAI Press Inc., 1989.

[ASE92] Noga Alon, Joel H. Spencer, and Paul Erdős. *The Probabilistic Method.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley and Sons, Inc., 1992.

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

[GW97] Oded Goldreich and Avi Wigderson. Tiny families of functions with random properties: A quality-size tradeoff for hashing. *Random Structures & Algorithms*, 11(4):315–343, 1997.

[GZ97] Oded Goldreich and David Zuckerman. Another proof that BPP $\subseteq$ PH (and more). *Electronic Colloquium on Computational Complexity* Technical Report TR97-045, September 1997. `http://www.eccc.uni-trier.de/eccc`.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 12–24, Seattle, Washington, 15–17 May 1989.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[Nis96] Noam Nisan. Extracting randomness: How and why: A survey. In *Proceedings, Eleventh Annual IEEE Conference on Computational Complexity*, pages 44–58, Philadelphia, Pennsylvania, 24–27 May 1996. IEEE Computer Society Press.

[NT98] Noam Nisan and Amnon Ta-Shma. Extracting randomness: A survey and new constructions. *Journal of Computer and System Sciences*, 1998. To appear in STOC '96 special issue. Preliminary versions in [Nis96] and [TS96].

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, October 1994.

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

[Pip87] Nicholas Pippenger. Sorting and selecting in rounds. *SIAM Journal on Computing*, 16(6):1032–1038, December 1987.

[RR99] Ran Raz and Omer Reingold. On recycling the randomness of the states in space bounded computation. These proceedings, 1999.

[RT97] Jaikumar Radhakrishnan and Amnon Ta-Shma. Tight bounds for depth-two superconcentrators. In *38th Annual Symposium on Foundations of Computer Science*, pages 585–594, Miami Beach, Florida, 20–22 October 1997. IEEE.

[Sip88] Michael Sipser. Expanders, randomness, or time versus space. *Journal of Computer and System Sciences*, 36(3):379–383, June 1988.

[SSZ98] Michael Saks, Aravind Srinivasan, and Shiyu Zhou. Explicit OR-dispersers with polylogarithmic degree. *Journal of the ACM*, 45(1):123–154, January 1998.

[STV98] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR lemma. Technical Report TR98-074, Electronic Colloquium on Computational Complexity, December 1998. Extended abstract in these proceedings.

[SV86] Miklos Santha and Umesh V. Vazirani. Generating quasi-random sequences from semi-random sources. *Journal of Computer and System Sciences*, 33(1):75–87, August 1986.

[SZ98] Aravind Srinivasan and David Zuckerman. Computing with very weak random sources. To appear in *SIAM Journal on Computing*, 1998. Preliminary version in *FOCS '94*.

[Tre98] Luca Trevisan. Constructions of near-optimal extractors using pseudo-random generators. *Electronic Colloquium on Computational Complexity* Technical Report TR98-55, September 1998. Extended abstract in these proceedings.

[TS96] Amnon Ta-Shma. On extracting randomness from weak random sources (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 276–285, Philadelphia, Pennsylvania, 22–24 May 1996.

[TS98a] Amnon Ta-Shma. Personal communication, August 1998.

[TS98b] Amnon Ta-Shma. Almost optimal dispersers. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 196–202, Dallas, TX, May 1998. ACM.

[Vaz84] Umesh V. Vazirani. *Randomness, Adversaries, and Computation*. PhD thesis, University of California, Berkeley, 1984.

[Vaz87a] Umesh V. Vazirani. Efficiency considerations in using semi-random sources (extended abstract). In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 160–168, New York City, 25–27 May 1987.

[Vaz87b] Umesh V. Vazirani. Strong communication complexity or generating quasirandom sequences from two communicating semirandom sources. *Combinatorica*, 7(4):375–392, 1987.

[VV85] Umesh V. Vazirani and Vijay V. Vazirani. Random polynomial time is equal to slightly-random polynomial time. In *26th Annual Symposium on Foundations of Computer Science*, pages 417–428, Portland, Oregon, 21–23 October 1985. IEEE.

[WZ95] Avi Wigderson and David Zuckerman. Expanders that beat the eigenvalue bound: Explicit construction and applications. Technical Report CS-TR-95-21, University of Texas Department of Computer Sciences, 1995. To appear in *Combinatorica*.

[Yao82] Andrew C. Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, 3–5 November 1982. IEEE.

[Zuc96] David Zuckerman. Simulating BPP using a general weak random source. *Algorithmica*, 16(4/5):367–391, October/November 1996.

[Zuc97] David Zuckerman. Randomness-optimal oblivious sampling. *Random Structures & Algorithms*, 11(4):345–367, 1997.