Extractors: Optimal up to Constant Factors

Chi-Jen Lu
Institute of Information Science,
Academia Sinica, Taipei, Taiwan.
cjlu@iis.sinica.edu.tw

Omer Reingold*
AT&T Labs - Research.
omer@research.att.com

Salil Vadhan†
Harvard University.
salil@eecs.harvard.edu.

Avi Wigderson‡
Institute for Advanced Study, Princeton and the
Hebrew University , Jerusalem.
avi@ias.edu

ABSTRACT

This paper provides the first explicit construction of extractors which are simultaneously optimal up to constant factors in *both* seed length and output length. More precisely, for every n, k, our extractor uses a random seed of length $O(\log n)$ to transform any random source on n bits with (min-)entropy k, into a distribution on $(1-\alpha)k$ bits that is ϵ-close to uniform. Here α and ϵ can be taken to be any positive constants. (In fact, ϵ can be almost polynomially small).

Our improvements are obtained via three new techniques, each of which may be of independent interest. The first is a general construction of *mergers* [22] from locally decodable error-correcting codes. The second introduces new *condensers* that have *constant seed length* (and retain a constant fraction of the min-entropy in the random source). The third is a way to augment the "win-win repeated condensing" paradigm of [17] with error reduction techniques like [15] so that the our constant seed-length condensers can be used without error accumulation.

*Address: AT&T Labs - Research, Room A201, 180 Park Avenue, Bldg. 103, Florham Park, NJ, 07932, USA. Part of this research was performed while visiting the Institute for Advanced Study, Princeton, NJ.

†Address: Harvard University, Division of Engineering and Applied Sciences, Maxwell Dworkin 337, 33 Oxford Street Cambridge, MA 02138, USA. URL: http://www.eecs.harvard.edu/~salil. Supported by NSF grant CCR-0133096 and a Sloan Research Fellowship.

‡Address: Institute for Advanced Study, School of Math., Einstein Drive, Princeton, NJ 08540.

STOC'03, June 9–11, 2003, San Diego, California, USA.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Random number generation, Probabilistic algorithms; G.2 [**Discrete Mathematics**]: Combinatorics, Graph Theory; F.2 [**Analysis of Algorithms and Problem Complexity**]: General; H.1.1 [**Systems and Information Theory**]: Information theory

General Terms

Theory, Algorithms

Keywords

Randomness Extractors, Pseudorandomness, Condensers, Mergers, Locally Decodable Error-Correcting Codes

1. INTRODUCTION

Extractors are functions that extract almost-uniform bits from sources of biased and correlated bits. Since their introduction by Nisan and Zuckerman [14], extractors have played a fundamental and unifying role in the theory of pseudorandomness. In particular, it has been discovered that they are intimately related to a number of other important and widely studied objects, such as hash functions [9], expander graphs [14, 28, 18, 23, 4], samplers [7, 30], pseudorandom generators [26] and error-correcting codes [26, 25, 24]. In addition, extractors have been found to have a vast and ever-growing collection of applications in diverse aspects of computational complexity, combinatorics, and cryptography. See the excellent surveys [13, 19].

Like the other objects listed above, extractors with very good parameters can be nonconstructively shown to exist via the Probabilistic Method, but finding *explicit* constructions — ones computable in polynomial time — has been much more difficult. A long body of work has sought to find explicit constructions which approach the optimal, nonconstructive bounds.

In this paper, we achieve one of the goals of this line of work, namely the explicit construction of extractors that are "optimal up to constant factors". In order to make sense of this, we need to define extractors more precisely.

1.1 Extractors and their parameters

To formalize the notion of extractors we first need a measure of randomness in a source of biased and correlated bits.

The definition offered by [5, 29], relies on min-entropy: A distribution X is a *k-source* if the probability it assigns to every element in its domain is at most 2^{-k}. The *min-entropy* of X, (denoted by $H_\infty(X)$) is the maximal k such that X is a k-source. We also need to define what an "almost uniform" distribution is. For random variables X,Y over a set S, we say that X and Y are *ε-close* if the statistical difference between X and Y is at most ε, i.e. for all $T \subseteq S$, $|\Pr[X \in T] - \Pr[Y \in T]| \leq \varepsilon$.Finally, denote by U_n the uniform distribution over $\{0,1\}^n$.

DEFINITION 1 (EXTRACTORS [14]). *A function* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \mapsto \{0,1\}^m$ *is a* (k,ε)-extractor *if for any k-source X over $\{0,1\}^n$, the distribution* $\mathrm{Ext}(X, U_d)$ *(the extractor's output on an element sampled from X and a uniformly chosen d bit string) is ε-close to U_m.*

In other words, Ext "extracts" m almost-uniform bits from a source with k bits of hidden randomness, using a random d-bit *seed* as a catalyst.[1] Naturally, the goal in constructing extractors is to simultaneously *maximize the output length* m (i.e., extract as much randomness as possible) and *minimize the seed length* d (i.e., investing as few truly random bits as possible). Another goal is to minimize the *error* ε. In this discussion we focus on the case that ε is an arbitrarily small constant, e.g. $\varepsilon = .01$.

Using the Probabilistic Method, it can be nonconstructively shown that for every n and $k \leq n$, there exist extractors with output length $m = k + d - O(1)$ and seed length $d = \log n + O(1)$, and these are tight up to an additive constant (when $k \leq n/2$) [14, 13]. That is, an *optimal extractor* extracts almost all the randomness in the source (and seed) using a seed of only logarithmic length.

1.2 Previous constructions

As mentioned earlier, a long line of work has sought to obtain explicit constructions whose parameters match that of the optimal extractor, or at least come within constant factors[2]. This goal has remained elusive despite a decade of effort, but there has been substantial progress towards it which, we briefly describe here. (For more detailed descriptions, see the surveys [13, 19].)

For the case of constant min-entropy rate, i.e. $k = \Omega(n)$, the construction of Zuckerman [30] is optimal within constant factors. That is, it achieves a seed length of $d = O(\log n)$ and output length $m = \Omega(k)$. Much of the subsequent work on extractors has aimed at matching these parameters for subconstant, indeed arbitrary, min-entropy rate. However, thus far, it was only known how to optimize one of two parameters to within a constant factor, while paying a superconstant factor in the other. Constructions with seed length $d = O(\log n)$ for arbitrary min-entropy were given in [26, 10, 17, 20], but the best output length achieved was $m = k/\log k$ [17, 23]. Constructions with output length $m = \Omega(k)$ were given in [13, 16, 17], but the best seed length achieved was $d = O(\log n \cdot \mathrm{polyloglog}\ n)$ [17].

To summarize, for *extractors* that work for *every min-entropy* k, it was only known how to optimize *either* the seed length *or* the output length to within a constant factor, but not both. In both cases, the best known construction was given by Reingold, Shaltiel, and Wigderson [17], who showed how to optimize either parameter while paying a polylogarithmic factor in the other.

1.3 New Results

In this paper, we construct extractors in which the seed length and output length are *simultaneously* optimal to within constant factors:

MAIN THEOREM. For any constants $\alpha, \varepsilon > 0$,[3] every n, and every $k \leq n$, there is an explicit (k,ε)-extractor $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, with $d = O(\log n)$ and $m = (1-\alpha) \cdot k$.

1.4 Techniques

We obtain our extractors by introducing three orthogonal new techniques, each of which may be of independent interest. In addition to these techniques, we rely on many ideas from [17, 15, 10, 27] and their predecessors.

Mergers from Locally Decodable Codes. Consider a random source X which consists of t blocks $X = (X_1, \ldots, X_t) \in (\{0,1\}^k)^t$ such that for some (unknown) i, X_i is uniform over $\{0,1\}^k$ and the other X_j's can depend arbitrarily on X_i. Such *somewhere random sources* were introduced in the work of Ta-Shma [13], and are clearly a special case of k-sources. Extractors for somewhere random sources are known as *mergers*.

Is it easier to construct mergers than general extractors? The significance of this question comes from the fact that good mergers imply good extractors for general sources. Indeed, Ta-Shma [13] showed how to convert a general source into a somewhere random sources (with polynomially many blocks). This is further improved to much fewer blocks (polylog or even polyloglog) with the repeated condensing of [17] and our improved version of it. Ta-Shma [13] constructed mergers with seed length $O(\log^2 n)$ (in fact $O((\log t) \cdot (\log k))$, which was indeed shorter than the best for general extractors at the time. However, this seed length is too large for the current march towards optimality.

In this paper, we give a completely new construction of mergers, based on locally decodable error-correcting codes. Informally, a *locally decodable code* is an error-correcting code such that given a codeword that has been corrupted in some fraction of positions, one can recover any individual symbol of the corresponding message by reading a very small (ideally constant) number of positions from the corrupted codeword. Locally decodable codes have received a lot of attention in the computer science literature, due in part to their applications to program testing, average-case complexity, and private-information retrieval (cf., [12, 3, 1, 21, 11]).

Roughly speaking, we show that any locally decodable

[1] The original definition of extractors in [14] is stronger than the one above (from [13]) in that it requires the seed to be explicitly included in the output. Our constructions also imply such strong extractors with almost the same parameters e.g. using the general transformation of [17].

[2] This was achieved for *dispersers*, which are a weaker relative of extractors which only require that the support of $\mathrm{Ext}(X, U_d)$ is of size at least $(1-\varepsilon) \cdot 2^m$. In [23] it was shown how to achieve $d = O(\log n)$ and $m = k$ for these objects

[3] In fact, the error can be as small as $\varepsilon = \exp(-\log n / \log^{(c)} n)$ for any constant c, where $\log^{(c)} n$ is the logarithm iterated c times. See Section 6 for our results for general ε.

code in which the decoding algorithm reads q codeword symbols to recover a single message symbol yields a merger which extracts a $1/q$ fraction of min-entropy from the source. (More precisely, it produces a string in $\{0,1\}^k$ of min-entropy k/q, so it actually only "condenses" the source.) Intuitively, we use the local decodability property to show that the source can be reconstructed from q-tuples of outputs of the merger. Since the source has k bits of min-entropy, this means that the outputs of the merger must have at least k/q bits of min-entropy (certainly on average, and even with high probability).

We believe this connection is of independent interest, and lends additional significance to the ongoing effort to close the huge gap between the upper and lower bounds for locally decodable codes (cf., [11]). In particular, a construction of locally decodable codes with polynomial rate and constant query complexity would yield a major simplification to our extractors.

At any rate, even with the currently available locally decodable codes, our new mergers use seed length which depends *only* on the *number* of blocks, but not on their length. This is used twice in the paper (explained in the next subsections). One use is conventional, extracting from a somewhere random source we create in our repeated condensing. Another is as a constant seed-length condenser used as a basic building block in the repeated condensing.

Optimal Repeated Condensing. A *condenser* converts a random source (with the help of a seed) into a new source of higher entropy rate (= min-entropy / source length) with high probability. A *lossless* condenser [23] is a special condenser whose output, while shorter than the input source, retains essentially[4] all of the min-entropy present in the source.

It is clear that a lossless condenser is desirable. We could repeat condensing until we get an extractor. In particular, if the seed has constant length and the output is a constant factor shorter than the input, then there is hope that such repeated condensing (with $O(\log n/k)$ iterations) would yield an extractor of optimal seed and output length — exactly what we are looking for.

There are at least two problems with this plan. One is how to obtain such a lossless condenser with constant seed length. We will deal with that in the next subsection, so assume for now we have one. The second problem is the error. Clearly, the error probability when using a constant-length seed must be constant as well. As we will have to repeat the condensing process more than a constant number of times, we cannot afford the error accumulation.[5]

One main contribution of this paper is a method for repeated condensing which is optimal in both seed length and error. Specifically, we can take any explicit lossless condenser as above and use it to construct, for every parameter s, an explicit lossless condenser, which uses $O(s)$ bits, and with probability at least $1-2^{-s}$ shrinks the source length by a factor of 2^s (of course, as long as the final output is larger than the entropy in the source).

Indeed, we have two different methods to achieve this task. In both we combine repeated condensing with an *error reduction* technique, in which we apply the condenser on several *dependent* seeds and concatenate the outputs. (This clearly costs us in the condensing factor, but it turns out to be affordable.) The first method of combining the two techniques applies repeated condensing until error goes almost up to 1, and then uses pairwise independent sampling to bring it down to the desired bound. A slightly similar technique is employed in the extractors of [10]. We omit this method due to space constraints. The second method, which we describe in detail, never allows the error to grow, by alternating condensing and error reduction via sampling seeds according to random edges in an expander graph, in the style of [15].

Constant Seed-Length Condensers and Win-Win Lossless Condensers. Now we return to the first problem: how to obtain such lossless condensers with constant seed length. Actually, we don't know how. However, it is nevertheless possible to pretend that we do, using the following win-win analysis, in the spirit of [10, 17].

The first step, which is in itself one of the key ideas of this paper, we construct a condenser with a constant seed length. However, in contrast to the lossless condensers above, these condensers only retain a constant fraction of the min-entropy of the source (with constant probability). We actually have two different constructions of such condensers, one using the mergers from locally decodable codes mentioned above, and another from appropriate sampling of sub-blocks of a (standard) error-corrected version of the source. We will not elaborate on these here.

Such condensers of course will not suffice for repeated condensing, due to the accumulated entropy loss (which does not happen in lossless condensers). However, if the output contains a constant fraction of the entropy, but not *all* of it, concatenating the input source to it immediately yields a what is known is *block source*: an object for which optimal extractors exist. (This is the win-win analysis of [17].) Thus we can employ the following strategy: We carry out the repeated condensing augmented with error reduction described above as if we have a lossless condenser. But at each application of the condenser, we also output a candidate block source. Thus, if we ever fail to condense losslessly, one of these candidates must actually be a block source (by the win-win analysis).

In order to formalize this, we abstract the notion of a *win-win condenser*, which outputs both a candidate condensed string and a set of candidate block sources. We carry out our recursion (repeated condensing + error reduction) on these win-win condensers, maintaining the invariant that either the condensing is lossless or one of the candidate block sources is indeed a block source. This is done until lossless condensing is impossible (the output length is shorter than the min-entropy), which means, by our invariant, that one of the candidates block source is indeed one. Extracting from all candidate block sources the same (optimal length) seed, we obtain a somewhere random source. Moreover, the number of candidates generated is small enough to allow us to use our new mergers.

1.5 Perspective

We stress that we do not consider our construction to

[4]There will be small entropy losses, but they will be insignificant

[5]The way [17] deal with that is indeed to use a larger (non-constant) seed, sufficient to make the error small enough to tolerate a union bound over all iterations of repeated condensing. Thus their extractors have suboptimal seed length

be *the* optimal extractor. First, achieving optimal error simultaneously with seed and output length remains a challenging problem (though we strongly believe it will yield soon). Next, finding a construction in which the output length and/or seed length are optimal to within *additive* constants remains an important open problem. For one, in some applications of extractors the relevant complexity measure is not the amount of min-entropy extracted, but rather the *min-entropy loss*, i.e. the number of bits *not* extracted. Clearly, optimizing the former to within a constant factor does not imply the same for the latter. In addition, in many applications, the complexity depends *exponentially* on the seed length d, so optimizing the seed to within an additive constant corresponds to optimizing the application to within a (multiplicative) constant factor. There has been significant progress towards individually optimizing the seed length and output length to within additive constants (cf., [16, 25]), but the price paid in the other parameter is more than a constant factor. Lastly, it would be nice to have a direct and self-contained construction, which does not require composing several components the way ours does. Indeed, our work suggests two appealing approaches to obtain a more direct construction. The first is to construct better mergers, which in turn would follow from a better construction of locally decodable codes. The second is to construct a *lossless* condenser with constant seed length, so that the win-win analysis described above is no longer needed.

2. PRELIMINARIES

For integer n, let $[n]$ denote the set $\{1, \ldots, n\}$. We use the convention that random variables are denoted by capital letters, whereas specific values for them are denoted by lowercase letters (e.g. $\Pr[X = x] = 1/2$). For a random variable X and an event E, we write $X|_E$ to denote the random variable X conditioned on E. For a distribution X, let $\mathrm{Supp}(X)$ denote the support of X. When we refer in this paper to an explicit function (such as an explicit extractor), this in fact refers to a *family* of functions that are computable in polynomial time (in their input length).

Block sources. One of the most useful notion in the extractor literature is that of a *block source* [5]. A block source is more structured than a general weak source:

DEFINITION 2. (X_1, X_2) *is a* (k_1, k_2)-block source *if* X_1 *is a* k_1*-source, and for every* $x_1 \in \mathrm{Supp}(X_1)$ *the distribution of* $X_2|_{X_1=x_1}$ *is a* k_2*-source.*

Intuitively, this means that X_2 contains k_2 bits of randomness that are "independent" of X_1. The task of extraction is made significantly easier given this additional structure of block sources. Indeed, good extractors for block-sources are already known. Such extractors are called block source extractors. Based on the method for block-source extraction of [14] and using newer extractors [16, 13, 10, 23, 20] it is possible to obtain a block-source extractors that extract k_1 bits using $O(\log n)$-long seed as long as k_2 is at least $\mathrm{poly}\log n$ (e.g., $\log^4 n$).

Somewhere Random Sources. A central notion in our paper is that of a *somewhere random source*, as introduced by Ta-Shma [13]. Roughly speaking, a somewhere random source is a random variable $X = (X_1, \ldots, X_b)$ where one of the X_i's is uniform and the others may depend arbitrarily on X_i. Actually, the definition allows the index i of the "good" block X_i to be a random variable depending on X. We will also use generalizations of this concept where the property required of the good block is not necessarily uniformity but some other property of sources:

DEFINITION 3. *Let* $\mathcal{C}$ *be a class of sources on* $\{0,1\}^m$*. A random variable* $X = (X_1, \ldots, X_b) \in (\{0,1\}^m)^b$ *is a* somewhere $\mathcal{C}$-source *if there is a random variable* $I = I(X)$ *taking values in* $[b]$ *such that for every* i*,* $X_i|_{I=i}$ *is in* $\mathcal{C}$ *(if* $\Pr[I = i] > 0$*).* I *is called the* selector *and* b *is called the number of* blocks.

If $\mathcal{C}$ *consists of only the uniform distribution on* $\{0,1\}^m$*, then* X *is called a* somewhere random source*. If* $\mathcal{C}$ *is the class of all* k*-sources on* $\{0,1\}^m$*, then* X *is called a* somewhere k-source*. If* $\mathcal{C}$ *is the class of all* (k_1, k_2) *block sources on* $\{0,1\}^m = \{0,1\}^{m/2} \times \{0,1\}^{m/2}$*, then* X *is called a* somewhere (k_1, k_2) block source.

In the above definition, we allow the selector I to be a probabilistic function of X.

Condensers. A condenser is a generalization of an extractor. Rather than requiring that it outputs a distribution that is (close to) uniform, the output distribution is only required to have at least some k' bits of randomness.

DEFINITION 4. *A* (k, k', ε)-condenser *is a function* $\mathrm{Con} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{n'}$*, such that for every* k*-source* X *of length* n*, the distribution* $\mathrm{Con}(X, U_d)$ *is* ε*-close to a* k'*-source.*

Note that an extractor is a special case of a condenser, when $n' = k'$.[6] It will also be useful to consider a special case of condensers where the output distribution is a somewhere random source.

DEFINITION 5. *A function* $\mathrm{Con} : \{0,1\}^n \times \{0,1\}^d \to (\{0,1\}^m)^b$ *is a* (k, ε)*-somewhere random condenser, if for every* k*-source* X *of length* n*, the distribution* $\mathrm{Con}(X, U_d)$ *is* ε*-close to a somewhere random source.*

Mergers. A *merger*, as defined by Ta-Shma [13], is a function which extracts randomness from somewhere random sources. That is, given a source consisting of t blocks $(X_1, \ldots, X_t)$, one of which is uniform, it "merges" the blocks to produce a single string Z which is close to uniform. We work with a generalization of this definition, whereby we only know that one of the blocks has high min-entropy and we only require that the output has high min-entropy. Thus the "progress" being made is that the output is shorter than the concatenation of the original t blocks, yet still contains much of the min-entropy. It will be convenient for us to consider mergers whose source and output are over general alphabets, though of course the notions of min-entropy and somewhere k-source are independent of the encoding.

DEFINITION 6. *A function* $\mathrm{Merge} : (\Sigma^n)^t \times \{0,1\}^d \to \Sigma^m$ *is a* (k, k', ε)-merger *if for every somewhere* k*-source* $X = (X_1, \ldots, X_t)$*, the random variable* $\mathrm{Merge}(X, U_d)$ *is* ε*-close to a* k'*-source.*

[6] We also note that in [17] condensers are defined with the additional requirement that, $k'/n' > k/n$. In other words, the condenser is required to improve the entropy rate (hence the name 'condenser'). We find it useful to define condensers more generally.

Finally, we note that good extractors for small values of k are already known (e.g., [10, 23]). This allows us to assume throughout the paper that k is large enough (i.e., at least $\mathrm{poly}\log n$).

3. MERGERS FROM LOCALLY DECODABLE CODES

In this section we give a general construction of mergers from locally decodable error-correcting codes. We consider this construction to be one of the main technical contributions of the paper. The major advantage of these mergers compared with previous constructions is that the seed length of the new mergers is independent of the *length* of the blocks. Instead it only depends on the *number* of blocks and the error parameter. This property turns out to be very useful. Indeed the extractors of this paper will be obtained by first transforming (using a seed of logarithmic length) an arbitrary k-source into a somewhere random source with "relatively few" blocks and then applying the new mergers on this somewhere random source. Since the somewhere random source has only few blocks, the seed length of the merger (and thus also the extractor) will be logarithmic. Another (more surprising) consequence of these mergers is that they directly imply condensers with constant seed length (see Section 5).

3.1 Locally Decodable Erasure Codes

Informally, a locally decodable code is an error-correcting code such that given a codeword that has been corrupted in some fraction of positions, one can recover any individual symbol of the corresponding message by reading very few positions of the codeword. Locally decodable codes have received a lot of attention in the computer science literature, due in part to their applications to program testing, average-case complexity, and private-information retrieval (cf., [12, 2, 21, 11]). The notion of locally decodable codes that we will use here differs in a couple ways from the standard ones.[7] First, we will only be concerned with decoding from *erasures* rather than corruptions. Second, we will not be concerned with the efficiency of the decoding procedure, except for the number of symbols read. We will not even require that the positions read by the decoder can be generated by some probabilistic sampling algorithm; instead we only require that the positions read depend on only the set of erased positions (as opposed to the actual symbols in the codeword).

DEFINITION 7. *Consider a code* $\mathrm{C} : \Sigma^t \to \Sigma^u$. *We say message position* $i \in [t]$ *is* decodable from *codeword positions* $(j_1, \ldots, j_q) \in [u]^q$ *if there is a function* $f : \Sigma^q \to \Sigma$ *such that for all* $x \in \Sigma^t$, $x_i = f(\mathrm{C}(x)_{j_1}, \ldots, \mathrm{C}(x)_{j_q})$. *We call* C *a* (q, ε) locally decodable erasure code *if for every* $S \subseteq [u]$ *of density greater than* ε *and every* $i \in [t]$, *there exists* $(j_1, \ldots, j_q) \in S^q$ *from which* i *is decodable.*

Intuitively, if all but an ε fraction of positions of a codeword are erased, then we can let S be the set of non-erased positions and the above definition guarantees that each message symbol can be recovered by reading only q non-erased symbols.

Ideally, we would like constructions of codes $\mathrm{C} : \Sigma^t \to \Sigma^u$ such that given $x \in \Sigma^t$ and $j \in [u]$, $\mathrm{C}(x)_j$ can be computed in time $\mathrm{poly}(\log |\Sigma|, t, \log u)$; we call such constructions *explicit*. If the computation time is instead $\mathrm{poly}(\log |\Sigma|, t, u)$, we call the construction *semi-explicit*.

The standard constructions of locally decodable codes satisfy Definition 7:

LEMMA 8 (HADAMARD CODES). *For every finite field* $\mathbb{F}$ *and* $t \in \mathbb{N}$, *there is an explicit* $(2, 1/|\mathbb{F}|)$ *locally decodable erasure code* $\mathrm{C} : \mathbb{F}^t \to \mathbb{F}^u$ *with* $u = |\mathbb{F}|^t$.

PROOF. For $x, y \in \mathbb{F}^t$, we define $\mathrm{C}(x)_y = \sum_i x_i y_i$. For $i \in [t]$, message position i is decodable from any pair of codeword positions (y, z) which disagree in only their i'th coordinates (because then $x_i = (\mathrm{C}(x)_y - \mathrm{C}(x)_z)/(y_i - z_i)$). Now suppose that $S \subset \mathbb{F}^t$ is such that i is not decodable from any pair of elements of S. This means that after fixing all entries of $y \in \mathbb{F}^t$ other than the i'th one, there is at most one setting of the i'th entry of y such that $y \in S$ (for otherwise S would contain two vectors disagreeing in only the i'th coordinate). Thus, S contains at most a $1/|\mathbb{F}|$ fraction of $\mathbb{F}^t$. ■

A similar argument applies for Reed–Muller codes of higher degree, using the fact that the value of a multivariate polynomial p of degree g at a point x can be computed from the values of p at any $d + 1$ points collinear with x.

LEMMA 9 (REED–MULLER CODES). *For every finite field* $\mathbb{F}$ *and* $g, m \in \mathbb{N}$, *there is a semi-explicit* $(g, g/|\mathbb{F}|)$ *locally decodable erasure code* $\mathrm{C} : \mathbb{F}^t \to \mathbb{F}^u$ *with* $t = \binom{m+g-1}{g-1}$ *and* $u = |\mathbb{F}|^m$.

3.2 Mergers

We first describe informally our construction of mergers from locally decodable codes. We view our somewhere random source $(X_1, \ldots, X_t) \in (\Sigma^n)^t$ as a $t \times n$ matrix X whose i'th row is X_i. We encode each *column* of this matrix using a locally decodable code, to obtain a bigger, $u \times n$ matrix $\hat{X}$. Our merger simply outputs a random row of this bigger encoded matrix.

Intuition for this construction can be obtained by considering the case of *Shannon* entropy. Suppose that X is a distribution over $t \times n$ matrices such that some (unknown) row of X has at least k bits of entropy; call this the *random row*. Then we will argue that at least a $1-\varepsilon$ fraction of rows of $\hat{X}$ have entropy at least (roughly) k/q.

Suppose not. This means that if we "erase" all rows of $\hat{X}$ which have entropy at least k/q, we are left with more than an ε fraction of the rows. But then, by local decodability of the error-correcting code, each row of X can be decoded from q unerased rows of $\hat{X}$. In particular, the random row of X can be described by q unerased rows of $\hat{X}$. But a string of entropy k cannot be described by q strings of entropy less than k/q.

Thus, the argument is a form of the "reconstruction paradigm" developed in [26, 23], whereby one proves that the output of an extractor or a condenser has high min-entropy by showing that otherwise the source would have a "small" description, contradicting the min-entropy assumed present in the

[7] Actually, despite these differences, it is shown in [6] that any code satisfying our definition also is also a code according to a more standard definition (with related parameters) and conversely, and thus lower bounds for standard locally decodable codes also apply to our definition.

source. Our construction shows how to benefit from the fact that, for somewhere random sources, we need only reconstruct a portion of the source to obtain a contradiction. For example, note that the parameters of the code we use, and hence the seed length of our merger, do not depend on n, the length of the blocks!

The following theorem describes the construction formally and states its properties in terms of min-entropy.

Theorem 10. *Given* $\mathrm{C} : \Sigma^t \to \Sigma^u$*, define* $\mathrm{Merge} : (\Sigma^n)^t \times [u] \to \Sigma^n$ *by* $\mathrm{Merge}((x_1, \ldots, x_t), j) = \mathrm{C}(y_1)_j \cdots \mathrm{C}(y_n)_j$*, where* $y_\ell = x_{1,\ell}x_{2,\ell} \cdots x_{t,\ell} \in \Sigma^t$*. If* C *is a* (q, ε) *locally decodable code, then for every* k *and* $\varepsilon > 0$*,* Merge *is a* $(k, k', 2\varepsilon)$ *merger for* $k' = (k - \log(1/\varepsilon))/q - \log u$.

We defer the proof of Theorem 10 (which follows the above intuition quite closely) to the full version. In the full version we also describe additional improvements (e.g. showing that the above are in fact "strong" mergers).

Now we describe the mergers obtained by applying Theorem 10 to the Hadamard and Reed-Muller codes. Using the Hadamard code over a field of size $O(1/\varepsilon)$, we get:

Corollary 11. *For every* $n, t \in \mathbb{N}$ *and* $\varepsilon > 0$*, there is an explicit* (k, k', ε) *merger* $\mathrm{Merge} : (\{0,1\}^n)^t \times \{0,1\}^d \to \{0,1\}^n$ *with* $d = O(t \cdot \log(1/\varepsilon))$ *and* $k' = k/2 - O(d)$.

Using the Reed–Muller code of degree $r+1$ over a field of size $O(r/\varepsilon)$, we get:

Corollary 12. *For every* $n, t \in \mathbb{N}$*,* $\varepsilon > 0$*, and* $m, r \in \mathbb{N}$ *such that* $\binom{m+r}{r} \geq t$*, there is an explicit* (k, k', ε) *merger* $\mathrm{Merge} : (\{0,1\}^n)^t \times \{0,1\}^d \to \{0,1\}^n$ *with* $d = O(m \cdot \log(r/\varepsilon))$ *and* $k' = \Omega(k/r) - O(d)$.

In particular, for an arbitrarily small constant δ*, we can have* $d = O(t^\delta \cdot \log(1/\varepsilon))$ *and* $k' = \Omega(k) - O(d)$*. Alternatively, we can have* $d = O(\log t)$ *and* $k' = \Omega(\varepsilon k/\log^2 t) - O(d)$ *for any* $\varepsilon \geq 1/t$.

We note that Reed–Muller codes are a central component in the recent extractor constructions of [23, 20]. Those constructions make use of additional algebraic structure of Reed–Muller codes beyond the local decodability property, and are concerned with extraction from general sources rather than exploiting the structure of somewhere random sources as we do.

4. WIN-WIN CONDENSERS

Win-win condensers, implicit in [17], are functions that take a weak random source X (and a short seed) and produce a shorter string X' together with a number of candidate block sources $(Z_1, \ldots, Z_b)$. The guarantee is that with high probability, either X' contains almost all of the min-entropy in X or one of the Z_i's will be a block source.

In this section, we give a new construction of win-win condensers. We obtain them by first constructing an initial win-win condenser with *constant seed length*. Then, by applying certain compositions to it, we improve its parameters and ultimately turn it into a somewhere random condenser. In a subsequent section, this is combined with our mergers from Section 3, to yield our final extractors.

4.1 Definition and Basic Properties

We begin by placing the notion of win-win condensers in a more general framework, where the candidate block sources can be replaced by somewhere $\mathcal{C}$-sources for any class $\mathcal{C}$ of random sources. It simplifies the compositions to work with a generalization of somewhere $\mathcal{C}$-sources where the selector gives a *set* of "good" indices rather than a single index.

Definition 13. *Let* $\mathcal{C}$ *be a class of sources on* $\{0,1\}^n$*. A random variable* $X = (X_1, \ldots, X_b) \in (\{0,1\}^n)^b$ *is a* generalized somewhere $\mathcal{C}$-source *if there is a random variable* $I = I(X)$ *taking values in* $2^{[b]}$ *such that for every* i*,* $X_i|_{i \in I}$ *is in* $\mathcal{C}$ *(if* $\Pr[i \in I] > 0$*).*

In this paper, we are most interested in the special case when $\mathcal{C}$ is the class of all (k_1, k_2) block sources on $\{0,1\}^m = \{0,1\}^{m/2} \times \{0,1\}^{m/2}$. (For notational convenience, we always pad our block sources with zeroes so that both blocks have the same length.) For this class (and other natural ones) generalized somewhere $\mathcal{C}$-sources are the same as standard somewhere $\mathcal{C}$-sources, up to a small loss in min-entropy:

Lemma 14. *Suppose* $X = (X_1, \ldots, X_b)$ *is a generalized somewhere* (k_1, k_2) *block source. Then* X *is a somewhere* $(k_1 - \log b, k_2 - \log b)$ *block source.*

The advantage of generalized somewhere $\mathcal{C}$-sources over standard somewhere $\mathcal{C}$-sources is captured by the following lemma.

Lemma 15. *Suppose* (X, Y) *is a random variable and* A *and* B *are events such that* $X|_A$ *is a generalized somewhere* $\mathcal{C}$*-source and* $Y|_B$ *is a generalized somewhere* $\mathcal{C}$*-source. Then* $(X, Y)|_{A \vee B}$ *is a generalized somewhere* $\mathcal{C}$*-source.*

Note that the above lemma does not require that A and B are disjoint (as seems necessary if we were to work with standard somewhere $\mathcal{C}$-sources).

Now we present our definition of win-win condensers. Intuitively, these are functions which either condense or produce a (generalized) somewhere $\mathcal{C}$-source.

Definition 16. *Let* $\mathcal{C}$ *be a class of sources on* $\{0,1\}^m$*. A pair of functions* $\langle \mathrm{Con}, \mathrm{Som} \rangle : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{\rho n} \times (\{0,1\}^m)^b$ *is called a* $(k, k', \varepsilon, \delta)$ win-win condenser with respect to $\mathcal{C}$ *if for every* k*-source* X*, there are subsets* $\textsc{con}, \textsc{som} \subseteq \{0,1\}^n \times \{0,1\}^d$ *such that if* Y *is uniform on* $\{0,1\}^d$*,*

1. $\mathrm{Con}(X, Y)|_{(X,Y) \in \textsc{con}}$ *is a* k'*-source.*
2. $\mathrm{Som}(X, Y)|_{(X,Y) \in \textsc{som}}$ *is a* b*-block generalized somewhere* $\mathcal{C}$*-source.*
3. *With probability at least* $1 - \delta$ *over* $x \stackrel{R}{\leftarrow} X$*,* $\Pr_Y[(x, Y) \in \textsc{con} \cup \textsc{som}] \geq 1 - \varepsilon$.

We say that $\langle \mathrm{Con}, \mathrm{Som} \rangle$ *has* seed error ε*,* source error δ*,* seed length d*,* condensation factor ρ*,* entropy loss $k - k'$*, and* b blocks.

Note the use of two error parameters in the definition. This is inspired by [15], where it was shown that the error of extractors can always be essentially separated into source and seed errors, where the source error can be made exponentially small simply by working with sources of slightly higher min-entropy. This allows error reduction to focus on the seed error (which is easier to handle because we can sample multiple seeds).

4.2 The initial win-win condenser

It turns out that good win-win condensers can be obtained from standard condensers, via the win-win analysis of [17]. Consider a condenser Con applied to a source X. The win-win analysis says that either (a) X still has some min-entropy[8] given $\mathrm{Con}(X, Y)$ and thus $(\mathrm{Con}(X, Y), X)$ forms a block source, or (b) $\mathrm{Con}(X, Y)$ retains almost all of the min-entropy in X. The benefit of this win-win analysis is that we can now iterate this condensing process on $\mathrm{Con}(X, Y)$, and only lose a small amount of min-entropy with each application (at the price of accumulating candidate block sources). We will actually combine this win-win analysis with the error analysis of [15] to make the source error of the win-win condenser much smaller than the error of the original condenser.

LEMMA 17. *Suppose* $\mathrm{Con} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a* (k, k', ε) *condenser. Define* $\mathrm{Som} : \{0,1\}^n \times \{0,1\}^d \to (\{0,1\}^{2n})^1$ *by* $\mathrm{Som}(x, y) = (\mathrm{Con}(x, y), x)$. *Then, for any* $\varepsilon \geq \delta > 0$ *and* $k_2 \in \mathbb{N}$, $\langle \mathrm{Con}, \mathrm{Som}\rangle$ *is a* $(k+\log(1/\delta), k-k_2-2\log(1/\delta), 4\varepsilon, 3\delta)$ *win-win condenser with respect to* $(k' - 3\log(1/\delta), k_2)$ *block sources.*

Proof Sketch: Let X be a $(k + \log(1/\delta))$-source, and let Y be uniform in $\{0,1\}^d$. Partition the elements $\{0,1\}^m$ according to their probability mass under $\mathrm{Con}(X, Y)$ as follows:

- $\textsc{lrg} = \{z : \Pr[\mathrm{Con}(X, Y) = z] > 2^{-(k'-\log(1/\varepsilon))}\}$.
- $\textsc{med} = \{z : 2^{-(k'-\log(1/\varepsilon))} \geq \Pr[\mathrm{Con}(X, Y) = z] > 2^{-(k-k_2)}\}$.
- $\textsc{sml} = \{z : 2^{-(k-k_2)} \geq \Pr[\mathrm{Con}(X, Y) = z]\}$.

The sets LRG, MED, SML roughly correspond to the cases of "error", "somewhere block source", and "condense". This is formalized in the following claims, which are proven in an analogous manner to corresponding statements in [15], [17], and [17], respectively.

CLAIM 18. *With probability at least* $1 - \delta$ *over* $x \overset{R}{\leftarrow} X$, $\Pr_Y[\mathrm{Con}(x, Y) \in \textsc{lrg}] \leq 2\varepsilon$.

CLAIM 19. *Suppose* $\Pr[\mathrm{Con}(X, Y) \in \textsc{sml}] > \delta^2$. *Then* $\mathrm{Con}(X, Y)|_{\mathrm{Con}(X,Y)\in\textsc{sml}}$ *is a* $(k - k_2 - 2\log(1/\delta))$*-source.*

CLAIM 20. *Suppose* $\Pr[\mathrm{Con}(X, Y) \in \textsc{med}] > \delta^2$. *Then* $(\mathrm{Con}(X, Y), X)|_{\mathrm{Con}(X,Y)\in\textsc{med}}$ *is a* $(k' - 3\log(1/\delta), k_2)$ *block source.*

If $\Pr[\mathrm{Con}(X, Y) \in \textsc{sml}] > \delta^2$, we set $\textsc{con} = \mathrm{Con}^{-1}(\textsc{sml})$, otherwise we set $\textsc{con} = \emptyset$. Similarly, if $\Pr[\mathrm{Con}(X, Y) \in \textsc{med}] > \delta^2$, we set $\textsc{som} = \mathrm{Con}^{-1}(\textsc{med})$, otherwise we set $\textsc{som} = \emptyset$. It is not difficult to verify that these settings satisfy all 3 conditions in the definition of win-win condensers. □

In Section 5, we will describe two simple constructions of condensers with a constant seed length. Applying the above lemma to them, we obtain the following win-win condensers with constant seed length. (Section 5 also sketches a direct construction of a win-win condenser with constant seed length.) We state the lemma in a more general form, and note that the seed length is constant for constant ε.

[8]We will denote this amount k_2 below, and it can be taken to be a fixed $\mathrm{poly}\log(n)$, and thus is negligible with respect to k, throughout the paper.

LEMMA 21. *For every* $\varepsilon > 0$, *every* constant $\rho > 0$, *every* $n, k \in \mathbb{N}$, *and every* $\delta \geq 2^{-\alpha k}$ *for some constant* $\alpha > 0$, *there exists an explicit* $(k, k - \Lambda, \varepsilon, \delta)$ *win-win condenser* $\langle \mathrm{Con}, \mathrm{Som}\rangle : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{\rho n} \times (\{0,1\}^n \times \{0,1\}^n)^b$ *with respect to* $(\Omega(k), k_2)$ *block sources, with*

- *condensation factor* ρ,
- *seed error* ε,
- *seed length* $d = O(\log(1/\varepsilon))$,
- *entropy loss* $\Lambda = O(k_2 + \log(1/\delta))$,
- *source error* δ, *and*
- $b = 1$ *block.*

4.3 Composing Win-Win Condensers

In this section, we describe two methods of composing win-win condensers which, when alternately applied to the initial win-win condenser above will yield our final win-win condenser. These composition methods actually apply to win-win condensers with respect to any class $\mathcal{C}$ of sources.

The first composition method, implicit in [17], is repeated condensing. This improves the condensation factor while paying a price in the other parameters (most significantly, the seed error).

LEMMA 22 (REPEATED CONDENSING). *Let* $\mathcal{C}$ *be any class of sources on* $\{0,1\}^m$. *Suppose we have an explicit* $(k, k - \Lambda, \varepsilon, \delta)$ *win-win condenser* $\langle \mathrm{Con}_1, \mathrm{Som}_1\rangle : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{\rho n} \times (\{0,1\}^m)^b$ *w.r.t.* $\mathcal{C}$, *and an explicit* $(k - \Lambda, k - 2\Lambda, \varepsilon, \delta)$ *win-win condenser* $\langle \mathrm{Con}_2, \mathrm{Som}_2\rangle : \{0,1\}^{\rho n} \times \{0,1\}^d \to \{0,1\}^{\rho^2 n} \times (\{0,1\}^m)^b$ *w.r.t.* $\mathcal{C}$.

Then there is an explicit $(k, k - \Lambda', \varepsilon', \delta')$ *win-win condenser* $\langle \mathrm{Con}, \mathrm{Som}\rangle : \{0,1\}^n \times \{0,1\}^{d'} \to \{0,1\}^{\rho' n} \times (\{0,1\}^m)^{b'}$ *w.r.t.* $\mathcal{C}$ *with*

- *condensation factor* $\rho' = \rho^2$,
- *seed error* $\varepsilon' = 3\varepsilon$,
- *seed length* $d' = 2d$,
- *entropy loss* $\Lambda' = 2\Lambda$,
- *source error* $\delta' = \delta + \delta/\varepsilon$, *and*
- $b' = 2b$ *blocks.*

Moreover, $\langle \mathrm{Con}, \mathrm{Som}\rangle$ *is computable in time* $\mathrm{poly}(n, d, m, b)$ *with one oracle query to* $\langle \mathrm{Con}_1, \mathrm{Som}_1\rangle$ *and one oracle query to* $\langle \mathrm{Con}_2, \mathrm{Som}_2\rangle$.

PROOF. For $x \in \{0,1\}^n$, $y_1, y_2 \in \{0,1\}^d$, we define

$$\mathrm{Con}(x, (y_1, y_2)) = \mathrm{Con}_2(\mathrm{Con}_1(x, y_1), y_2)$$

and

$$\mathrm{Som}(x, (y_1, y_2)) = (\mathrm{Som}_1(x, y_1), \mathrm{Som}_2(\mathrm{Con}_1(x, y_1))).$$

We defer the analysis to the full version.

■

The second composition reduces the seed error while paying a price the other parameters (most importantly, the condensation factor). Since we have already separated the source and seed errors as in [15], this error reduction is quite easily done by running the win-win condenser on multiple seeds (generated by a randomness-efficient sampler with an appropriate hitting property). Following [15], we generate a pair of dependent seeds using a random edge in an expander graph, because this gives good parameters for the subsequent recursion. However, other methods of sampling several seeds (such as pairwise independence) can also be analyzed in a similar fashion.

LEMMA 23 (REDUCING THE SEED ERROR). *Let $\mathcal{C}$ be any class of sources on $\{0,1\}^m$. Suppose there is an explicit $(k, k-\Lambda, \varepsilon, \delta)$ win-win condenser $\langle \mathrm{Con}, \mathrm{Som}\rangle : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{\rho n} \times (\{0,1\}^m)^b$ w.r.t. $\mathcal{C}$. Then there is an explicit $(k, k-\Lambda', \varepsilon', \delta')$ win-win condenser $\langle \mathrm{Con}', \mathrm{Som}'\rangle : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{\rho' n} \times (\{0,1\}^m)^{b'}$ w.r.t. $\mathcal{C}$ with*

- *condensation factor $\rho' = 2\rho$,*
- *seed error $\varepsilon = 2\varepsilon^2$,*
- *seed length $d' = d + O(\log(1/\varepsilon))$,*
- *entropy loss $\Lambda' = \Lambda + 1$,*
- *source error $\delta' = \delta$, and*
- *$b' = 2b$ blocks.*

Moreover, $\langle \mathrm{Con}', \mathrm{Som}'\rangle$ is computable in time $\mathrm{poly}(n, d, m, b)$ *with two oracle queries to $\langle \mathrm{Con}, \mathrm{Som}\rangle$.*

PROOF. As in [15], we use two dependent seeds with an appropriate hitting property. Specifically, let $\langle \mathrm{H}_1, \mathrm{H}_2\rangle : \{0,1\}^{d'} \to \{0,1\}^d \times \{0,1\}^d$ be a function such that for every set $S \subseteq \{0,1\}^d$ of density $\geq 1-\varepsilon$, if we choose Z uniformly in $\{0,1\}^{d'}$, we have

1. $\mathrm{H}_1(Z)$ and $\mathrm{H}_2(Z)$ are individually uniform in $\{0,1\}^d$, and
2. $\Pr[\mathrm{H}_1(Z) \in S \vee \mathrm{H}_2(Z) \in S] \geq 1 - 2\varepsilon^2$.

Such a function can be obtained explicitly with $d' = d + O(\log(1/\varepsilon))$ by taking a random edge on a sufficiently good expander or (almost equivalently) by using the high min-entropy extractors of [8].

We define $\mathrm{Con}'(x,z) = \mathrm{Con}(x, \mathrm{H}_1(z)) \circ \mathrm{Con}(x, \mathrm{H}_2(z))$ and $\mathrm{Som}'(x,z) = (\mathrm{Som}(x, \mathrm{H}_1(z)), \mathrm{Som}(x, \mathrm{H}_2(z)))$. Let X be a k-source on $\{0,1\}^n$ and let $\textsc{con}, \textsc{som} \subseteq \{0,1\}^n \times \{0,1\}^d$ be the sets guaranteed by the win-win property of Con. Define $\textsc{con}' = \{(x,z) : (x, \mathrm{H}_1(z)) \in \textsc{con} \vee (x, \mathrm{H}_2(z)) \in \textsc{con}\}$ and $\textsc{som}' = \{(x,z) : (x, \mathrm{H}_1(z)) \in \textsc{som} \vee (x, \mathrm{H}_2(z)) \in \textsc{som}\}$. It is not difficult to verify that these satisfy the definition of win-win condenser with the stated parameters, using the hitting property of H for the seed error and using Lemma 15 for $\textsc{som}$. ■

4.4 The Recursion

According to Lemma 22 and Lemma 23, if we compose a win-win condenser with itself and then apply error reduction, we can essentially square the condensation factor at the price of doubling the seed length while still maintaining the seed error. We will apply this process recursively for $r = O(\log\log(n/k))$ iterations until no further condensing is possible. It is easy to show the following by induction.

LEMMA 24. *Let $\mathcal{C}$ be any class of sources on $\{0,1\}^m$. Suppose for small enough ε_0 (say $\varepsilon_0 \leq 1/18$) there is an explicit $(k, k-\Lambda_0, \varepsilon_0, \delta_0)$ win-win condenser $\langle \mathrm{Con}_0, \mathrm{Som}_0\rangle : \{0,1\}^n \times \{0,1\}^{d_0} \to \{0,1\}^{\rho_0 n} \times (\{0,1\}^m)^{b_0}$ w.r.t. $\mathcal{C}$. Then for any $r \in \mathbb{N}$, there is an explicit $(k, k-\Lambda_r, \varepsilon_r, \delta_r)$ win-win condenser $\langle \mathrm{Con}_r, \mathrm{Som}_r\rangle : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{\rho_r n} \times (\{0,1\}^m)^{b_r}$ w.r.t. $\mathcal{C}$, with*

- *condensation factor $\rho_r = (2\rho_0)^{2^r}$, seed error $\varepsilon_r = \varepsilon_0$, seed length $d_r = 2^r(d_0 + O(\log(1/\varepsilon_0)))$, entropy loss $\Lambda_r = 2^r O(\Lambda_0)$, source error $\delta_r = (2/\varepsilon_0)^r \delta_0$, and $b_r = 4^r b_0$ blocks.*

Let $\langle \mathrm{Con}_0, \mathrm{Som}_0\rangle$ be the win-win condenser in Lemma 21, with $\delta_0 < (\varepsilon_0/2)^{-\log\log(n/k)}$ and small enough ρ_0, ε_0. Then we can choose $r < \log\log(n/k)$ such that

- $\rho_r n < k/2$, $\varepsilon_r = \varepsilon_0$, $d_r = O(\log(n/k)\log(1/\varepsilon_0))$, $\Lambda_r = \mathrm{poly}\log(n/\varepsilon_0) < k/2$,[9] $\delta_r \leq \varepsilon_0$, and $b_r \leq \log^2(n/k)$.

Since $\rho_r n < k/2 < k - \Lambda_r$, condensing is impossible. As a result, we get from Som_r a distribution which is $2\varepsilon_0$-close to a $\log^2(n/k)$-block generalized somewhere $(\Omega(k), k_2)$ block-source, which is a somewhere $(\Omega(k) - \log^2(n/k), k_2 - \log^2(n/k))$ block-source by Lemma 14. Applying efficient block-source extractors (see Section 2), on each one of the candidate block-sources using the same seed gives a somewhere random source.

THEOREM 25. *For any $n, k \in \mathbb{N}$ and any $\varepsilon \in (0,1)$, there are an explicit (k, ε)-somewhere random condensers* Con : *$\{0,1\}^n \times \{0,1\}^d \to (\{0,1\}^m)^b$, with output length $m = \Omega(k)$ and*

1. *seed length $d = O(\log n + \log(n/k) \cdot \log(1/\varepsilon))$ and $b =$* $\mathrm{poly}\log(n/k)$*, or*
2. *seed length $d = O(\log(n/\varepsilon))$ and $b = \mathrm{poly}\log(n/k) \cdot \log(1/\varepsilon)$ blocks.*

Part 2 is obtained by by taking Part 1 with constant ε and then applying Lemma 23 loglog$(1/\varepsilon')$ times to achieve any error ε' (at the price of producing slightly more blocks).

REMARK 26. The seed length $O(\log n) + O(\log(n/k)\log(1/\varepsilon))$ in Part 1 has two terms. The first term of $O(\log n)$ comes from the block extraction, and can be made as small as $(1 + o(1))\log n$ based on block extractors that rely on [25, 20]. This means that the entire seed length above can be made to be $(1+o(1))\log n$ in the case where $k = n^{(1-o(1))}$. We will elaborate on this feature in the final version.

5. CONDENSERS: CONST. SEED LENGTH

In this section we describe two completely different methods of constructing condensers with seed length as small as constant. The particular parameters needed for the construction of extractors are given by the following lemma.

LEMMA 27. *For every* constant $\rho > 0$*, every $\varepsilon > 0$, and every $n, k \in \mathbb{N}$, there exists an explicit $(k, \Omega(k), \varepsilon)$ condenser* Con : *$\{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{\rho n}$ with seed length $d = O(\log(1/\varepsilon))$.*

As we consider these condensers to be of independent interest, we describe them for a more general range of parameters than that of Lemma 27. We also discuss a direct construction of (almost lossless) win-win condensers based on error correcting codes.

[9] Recall that we assume k being larger than any $\mathrm{poly}\log(n/\varepsilon_0)$, because otherwise the desired extractor has been known already, as discussed at the end of Section 2.

5.1 Condensers from mergers

The first method of constructing condensers is based on a very useful observation suggested to us by Ran Raz. The observation is that *any merger can also be viewed as a condenser.*

Let X be any k-source. For any parameter t, divide X arbitrarily into t parts $X = (X_1, \ldots, X_t)$. It is not difficult to show that X is $t \cdot 2^{-\Omega(k/t)}$-close to a somewhere $k/2t$-source. Therefore any merger is indeed also a condenser:

LEMMA 28. *Let* Merge : $(\{0,1\}^n)^t \times \{0,1\}^d \to \{0,1\}^n$ *be a* $(k/2t, k', \varepsilon)$ *merger then* Merge *is also a* $(k, k', \varepsilon + t \cdot 2^{-\Omega(k/t)})$ *condenser.*

We therefore get that the mergers of Section 3 directly imply interesting condensers. In particular, from Corollary 11 we can deduce the following.

COROLLARY 29. *For every* $n, k, t \in \mathbb{N}$ *and* $\varepsilon > 0$*, there exists an explicit* $(k, k', \varepsilon + t \cdot 2^{-\Omega(k/t)})$ *condenser* Con : $\{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{\lceil n/t \rceil}$ *with* $d = O(t \cdot \log(1/\varepsilon))$ *and* $k' = \Omega(k/t) - O(d)$.

The above Corollary easily implies Lemma 27.

5.2 Encode-then-sample condensers

We now turn to the second method of constructing condensers. This will give a different proof of Lemma 27. The condensers described here follow ideas from [14, 17, 27] with an interesting twist of our own. The parameters obtained by these condensers (given below in Lemma 30) also imply Lemma 27. In addition, the seed length of these condensers has better dependence on the condensation factor $1/t$ and the error ε than the dependance obtained in Corollary 29.

LEMMA 30. *For some fixed polynomial* poly*, for every* $n, k, t \in \mathbb{N}$ *and* $\varepsilon > 0$ *such that* $k > \mathrm{poly}(t, \log 1/\varepsilon)$*, there exists an explicit* (k, k', ε) *condenser* Con : $\{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{\lceil n/t \rceil}$ *with* $d = O(\log(t/\varepsilon))$ *and* $k' = \Omega(k/t)$.

We coin the condensers used to prove Lemma 30 "encode-then-sample" as these condensers operate as follows: (1) Encode the input x with a (constant rate and constant relative distance) error correcting code EC. (2) Sample part of the encoded string $\bar{x} = \mathrm{EC}(x)$ and output it. For the second step we use a sampler taken from [27] which first evenly divides $\bar{x}$ into $\ell = O(t \log 1/\varepsilon)$ blocks $\bar{x}(1) \ldots \bar{x}(\ell)$. Using $d = O(\log(t/\varepsilon))$ bits, the sampler then selects a subset of $\ell/(c \cdot t)$ blocks where c is the rate of EC. The output of the condenser is the concatenation of these $\ell/(c \cdot t)$ blocks. We defer further details to the full version.

5.3 Win-win condensers from codes

Here we briefly discuss a very simple win-win condenser based on error correcting codes. We consider these condensers interesting for two reasons. First they directly give condensation that can be larger than constant (rather than starting with constant condensation and enlarging it through repeated condensing and error reduction). Even more interestingly, these condensers give new understanding regarding the nature of error corrected sources. The relation of error correcting codes to weak random sources has been the subject of intense interest since Trevisan [26] demonstrated the usefulness of error correcting weak random sources.

Consider a k-source X and an error correcting code EC with constant rate c and constant relative distance. As mentioned above, a random substring of $\bar{X} = \mathrm{EC}(X)$ of $O(k)$ bits is close to a k-source. In this sense an error corrected weak random source is "similar" to a k-wise independent source (i.e. one in which every k bits are random and independent). However, unlike the latter (where every substring contains all of the entropy), it is not easy to find a short substring of $\bar{X}$ which is close to a k-source. At the moment, the most randomness efficient way of selecting such "good" substrings is using combinatorial designs [26, 15]. It is interesting to analyze the entropy of more natural substrings.

Divide $\bar{X}$ to $t > 2c$ blocks $\bar{X}(1) \ldots \bar{X}(t)$. What can we say regarding the entropy of the blocks? As it turns out, each individual block may contain *no entropy.* Furthermore, it is possible to design X and EC such that each one of the blocks will only contain $O(k/t)$ bits of entropy. This example indeed illustrates how non-similar error corrected sources and k-wise independent sources can really be. Nevertheless, based on the proof of Lemma 30 and a refined win-win analysis we are able to prove an interesting claim regarding these blocks. Essentially, either one of these blocks contains $k - O(k_2)$ bits of entropy or one of a list of 2^t candidates is an $(\Omega(k), k_2)$ block source. For every set $S \subset [t]$ a candidate block source is $(\bar{X}(S), X)$ where $\bar{X}(S)$ is the concatenation of the blocks $\bar{X}(i)$ for $i \in S$. For lack of space we defer a more formal statement and a proof to the full version. We note though that this directly implies a win-win condenser $\langle \mathrm{Con}, \mathrm{Som} \rangle$ with condensation factor c/t and seed length $\log t$, where the min-entropy in the block sources remains $\Omega(k)$ even when t is nonconstant. For $i \in [t]$ $\mathrm{Con}(x, i)$ is simply defined as $\bar{x}(i)$, and $\mathrm{Som}(x, i)$ just ignores i and outputs the 2^t candidate block sources.

6. PUTTING IT TOGETHER

We know from Theorem 25 that for any ε, one can use a seed of length $O(\log n + \log(n/k)\log(1/\varepsilon))$ to produce a distribution that is ε-close to a generalized somewhere random source of $\mathrm{poly}\log(n/k)$ blocks of length $m = \Omega(k)$. Next, apply on it our (m, k', ε) Reed-Muller merger in Corollary 12 with $k' = \Omega(m) = \Omega(k)$ using a seed of length $O(\log(n/k)\log(1/\varepsilon))$. The result is a distribution that is 2ε-close to an $\Omega(k)$-source of length $m \leq k$, on which we can apply Zuckerman's extractor [30] to extract $\Omega(k)$ bits. We can iterate the extraction using the technique of [28] to extract all but a small constant fraction of randomness. Then we have the following.

LEMMA 31. *For any constant* $\alpha \in (0,1)$*, every* $n \in \mathbb{N}$ *and* $k \leq n$*, and every* $\varepsilon \in (0,1)$ *where* $\varepsilon > \exp(-k/2^{O(\log^* k)})$*,*[10] *there is an explicit* (k, ε)*-extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^{O(\log n + \log(n/k) \cdot \log(1/\varepsilon))} \to \{0,1\}^{(1-\alpha)k}$.

For constant ε, this gives our main extractor. It is interesting to note that this extractor is derived by starting from a condenser with constant seed length and then applying the repeated condensing (using Lemma 22 and Lemma 23) as described in the introduction.

THEOREM 32. *For any constants* $\alpha, \varepsilon \in (0,1)$*, every* $n \in \mathbb{N}$ *and* $k \leq n$*, there is an explicit* (k, ε)*-extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^{O(\log n)} \to \{0,1\}^{(1-\alpha)k}$.

[10]This restriction on ε is inherited from an analogous restriction in [30].

General ε. In Theorem 32, both the seed length and the output length are optimal up to constant factors, but the error is not. A possible solution is to use the error reduction method for extractors of [15] (as is done in [17]). However, this will give us extractors with seed length $O(\log n \cdot \operatorname{poly}\log\log n + \log(1/\varepsilon))$, which loses the entire advantage of our extractors. The problem is that the error reduction of [15] is suboptimal when applied to constant (or high) error extractors. Our extractor in Lemma 31 allows us to improve this range of the error reduction of [15] (using ideas and results from both [15] and [17]). Thus we also obtain (k,ε)-extractors for an arbitrary ε, and with seed length and output length that are simultaneously "near-optimal". For space constraints we defer the details of the general error reduction theorem to the full version. Applying it to Theorem 32, we get our final extractors.

THEOREM 33. *For any constant $\alpha \in (0,1), c \in \mathbb{N}$, for every k, and every $\varepsilon \in (0,1)$ where $\varepsilon > \exp(-k/2^{O(\log^* k)})$, there are explicit (k,δ)-extractors* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, *with each one of the following parameters:*

1. $d = O(\log n)$, $m = (1-\alpha)k$, *and* $\delta = (1/n)^{1/\log^{(c)} n}$.
2. $d = O((\log^* n)^2 \log n + \log(1/\delta))$, $m = (1-\alpha)k$, *and* $\delta = \varepsilon$.
3. $d = O(\log(n/\delta))$, $m = \Omega(k/\log^{(c)} n)$, *and* $\delta = \varepsilon$.

Acknowledgements

We are grateful to Ran Raz for many fruitful discussions. Particularly, we would like to thank him for pointing out to us the connection between mergers and condensers discussed in Section 5. We would also like to thank Ronen Shaltiel for very useful discussions.

7. REFERENCES

[1] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. *Comput. Complexity*, 1(1):3–40, 1991.

[2] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In *Proc. 7th STACS*, 1990.

[3] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993.

[4] M. Capalbo, O. Reingold, S. Vadhan, and A. Wigderson. Randomness conductors and constant-degree lossless expanders. In *Proc. 34th STOC*, pages 659–668, 2002.

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

[6] R. de Wolf and I. Kerenidis. Exponential lower bounds for 2-query locally decodable codes. In *these proceedings*, 2003.

[7] O. Goldreich. A sample of samplers: A computational perspective on sampling. Technical Report TR97-020, Elec. Colloq. on Comput. Complexity, May 1997.

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

[9] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

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

[11] J. Katz and L. Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proc. 32nd STOC*, pages 80–86, 2000.

[12] R. Lipton. New directions in testing. In *Proc. DIMACS Wkshp. on Dist. Comput. & Crypto.*, 1989.

[13] N. Nisan and A. Ta-Shma. Extracting randomness: A survey and new constructions. *J. Comput. Syst. Sci.*, 58(1):148–173, 1999.

[14] N. Nisan and D. Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, Feb. 1996.

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

[16] R. Raz, O. Reingold, and S. Vadhan. Extracting all the randomness and reducing the error in Trevisan's extractors. *J. Comput. Syst. Sci.*, 65(1):97–128, 2002.

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

[18] O. Reingold, S. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *Proc. 41st FOCS*, pages 3–13, 2000.

[19] R. Shaltiel. Recent developments in explicit constructions of extractors. *Bull. EATCS*, 77, 2002.

[20] R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In *Proc. 42nd FOCS*, pages 648–657, 2001.

[21] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. Syst. Sci.*, 62:236–266, 2001.

[22] A. Ta-Shma. On extracting randomness from weak random sources (extended abstract). In *Proc. 28th STOC*, pages 276–285, 1996.

[23] A. Ta-Shma, C. Umans, and D. Zuckerman. Loss-less condensers, unbalanced expanders, and extractors. In *Proc. 33rd STOC*, pages 143–152, 2001.

[24] A. Ta-Shma and D. Zuckerman. Extractor codes. In *Proc. 33rd STOC*, pages 193–199, 2001.

[25] A. Ta-Shma, D. Zuckerman, and S. Safra. Extractors from Reed–Muller codes. In *Proc. 42nd FOCS*, 2001.

[26] L. Trevisan. Extractors and pseudorandom generators. *J. ACM*, 48(4):860–879, July 2001.

[27] S. P. Vadhan. On constructing locally computable extractors and cryptosystems in the bounded storage model. Cryptology ePrint Arch., 2002/162, 2002.

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

[29] D. Zuckerman. Simulating BPP using a general weak random source. *Algorithmica*, 16(4/5), 1996.

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