Extractor Codes

Amnon Ta-Shma, David Zuckerman

Abstract—**We study error correcting codes for highly noisy channels. For example, every received signal in the channel may originate from some half of the symbols in the alphabet. Our main conceptual contribution is an equivalence between error correcting codes for such channels and extractors. Our main technical contribution is a new explicit error correcting code based on Trevisan's extractor that can handle such channels, and even noisier ones. Our new code has polynomial-time encoding and polynomial-time soft-decision decoding. We note that Reed-Solomon codes cannot handle such channels, and our study exposes some limitations on list decoding Reed-Solomon codes.**

Another advantage of our equivalence is that when the Johnson bound is restated in terms of extractors, it becomes the well-known Leftover Hash Lemma. This yields a new proof of the Johnson bound which applies to large alphabets and soft decoding.

Our explicit codes are useful in several applications. First, they yield algorithms to extract many hardcore bits using few auxiliary random bits. Second, they are the key tool in a recent scheme to compactly store a set of elements in a way that membership in the set can be determined by looking at only one bit of the representation. Finally, they are the basis for the recent construction of high-noise, almost optimal rate list-decodable codes over large alphabets [1].

Keywords— **Extractors, extractor codes, hardcore bits, Johnson bound, list decoding, Reed-Solomon codes, soft-decision decoding.**

I. Introduction

Consider a channel over a large alphabet Σ such that upon sending $(\sigma_1, \ldots, \sigma_{\mathrm{T}}) \in \Sigma^{\mathrm{T}}$, the receiver is only able to determine sets $\mathcal{S}_i \subseteq \Sigma$, $|\mathcal{S}_i| = \frac{|\Sigma|}{2}$, such that $\sigma_i \in \mathcal{S}_i$. Can we find codes for which the number of codewords $(v_1, \ldots, v_{\mathrm{T}})$ such that for all i, $v_i \in \mathcal{S}_i$, is small? What if we require $v_i \in \mathcal{S}_i$ for only 51% of i?

This scenario has been raised before in the context of list decoding. If we allow more than half the distance d errors, then guaranteed unambiguous decoding is impossible. List decoding tries to rectify this situation. We give up on guaranteed unambiguous decoding and instead we require that the number of codewords with modest agreement with the received word is small (and in the algorithmic version we output all these codewords). Thus, an adversary choosing $\frac{d}{2}$ errors may force ambiguous decoding, but will never be able to make this ambiguity large. A useful view of this is that the number of codewords in any large ball is small.

Although list decoding was defined independently by Elias [2] and Wozencraft [3] in the 1950's, no non-trivial list decoding algorithm was known until the late 1980's. Since then, there have been several [4], [5], [6], [7]. For an excellent survey paper and for the history of the list decoding problem, we refer the reader to [8], and Section 3.3 therein in particular.

As it turns out, the list decoding algorithms above deal also with the case where for each i the receiver has a *small* set $\mathcal{S}_i$ of possible explanations of the i'th symbol. In fact, Guruswami and Indyk [9] subsequently called the natural generalization of this property "list-recoverable," and showed how to use this property of Reed-Solomon codes to build better list decodable codes. A natural question is then, can one find such a code for the case where the sets $\mathcal{S}_i$ are large?

We first show that when the sets $\mathcal{S}_i$ are large, Reed-Solomon codes could have exponentially many codewords such that for all i, $v_i \in \mathcal{S}_i$. Thus, Reed-Solomon codes are not a good choice for such a scenario. In contrast, we exhibit codes with only polynomially many codewords having 51% of the $v_i \in \mathcal{S}_i$, even when the errors are picked adversarially. This property also holds in the soft-decision model. We give explicit codes that have polynomial-time encoding and list decoding, and non-explicit codes with better parameters.

Our codes are based upon a combinatorial object from the study of pseudorandomness called an extractor. An extractor is a procedure that extracts randomness from a defective random source, using a small additional number of truly random bits. Extractors were first defined and constructed in [10], and have since been improved by several authors. Extractors have been used to construct various types of pseudorandom generators ([10], [11], [12]), and have had important applications in seemingly unrelated areas, including expander graph and superconcentrator constructions [13], time-space tradeoffs [14], and unapproximability [15], [16]. See [17], [18], [19] for surveys of extractors.

This paper gives yet another unexpected application of extractors. We show an equivalence between extractors and codes which decode in highly noisy channels in the adversarial model. This equivalence implies bounds on the parameters such codes must have. For instance, the rate of such codes must be smaller than $\frac{1}{\log|\Sigma|}$, and there is a word with at least $\Omega(\frac{|\Sigma|}{n})$ close codewords.

This equivalence also sheds light on the Johnson bound [20]. Restated in terms of extractors, the Johnson bound becomes the well-known Leftover Hash Lemma. This yields

A preliminary version of this paper appeared in *Proceedings of the 33rd ACM Symposium on Theory of Computing*, 2001, pp. 193–199.

A. Ta-Shma is with the Computer Science Department, Tel-Aviv University, Tel-Aviv, Israel. E-mail: amnon@post.tau.ac.il. Some of this work was done while the author was at the University of California at Berkeley, and supported in part by a David and Lucile Packard Fellowship for Science and Engineering and NSF NYI Grant No. CCR-9457799.

D. Zuckerman is with the Computer Science Department, University of Texas, Austin, Texas. E-mail: diz@cs.utexas.edu. Some of this work was done while the author was at the University of California at Berkeley. Supported in part by a David and Lucile Packard Fellowship for Science and Engineering and NSF Grant CCR-9912428, NSF NYI Grant No. CCR-9457799, and an Alfred P. Sloan Research Fellowship.

a new proof of the bound which applies to large alphabets and soft decoding.

Our explicit codes are based on Trevisan's explicit extractor [21]. We show that it has polynomial-time encoding, polynomial-time list-decoding, and is capable of handling the above noisy channel even in the adversarial model. Thus, we build an explicit code that has better soft-decision list decoding than Reed-Solomon, and can tolerate much more noise.

These explicit codes can be used to extract hardcore bits, a very useful tool in cryptography and the earliest application of list decoding in computer science [4]. We are able to extract many hardcore bits using few auxiliary random bits.

II. Top down overview

A. Soft-Decision Decoding

In reality, channel noise is almost always a continuous phenomenon. Thus, on receiving a signal σ the detector may decide that the transmitted symbol was a_1 with probability $\frac{98}{100}$, a_2 with probability $\frac{1}{100}$ and a_3 with probability $\frac{1}{100}$. A hard-decision detector may transform the signal σ to the symbol a_1. Soft-decision decoding tries to use these probabilities.

Early attempts at soft-decision decoding start with Forney's GMD decoding [22]. Recent soft decoding algorithms include [23] for Reed-Solomon codes and [24] for Chinese Remainder codes. Some of these papers (e.g., [24]) study a soft-decision detector that outputs a single Σ symbol along with a confidence level, thus using only some of the information the channel contains. We now formally define soft decoding.

When we receive a symbol from a noisy channel, we can infer a probability distribution over all possible symbols in the alphabet $\Sigma = [\mathrm{M}]$. We deal with time-varying channels; seeing a symbol $\sigma \in [\mathrm{M}]$ at time $i \in \mathrm{T}$ induces a weight function $w_i : [\mathrm{M}] \to [0,1]$, where $w_i(z)$ corresponds to the probability that the i'th transmitted symbol is z given that the i'th received symbol is σ.

In fact, we can also deal with noisy channels where seeing the whole sequence $\sigma_1, \ldots, \sigma_{\mathrm{T}} \in [\mathrm{M}]^T$ induces a *product* weight function $w : [\mathrm{T}] \times [\mathrm{M}] \to [0,1]$, where $w(i,z)$ corresponds to the probability that the i'th transmitted symbol is z when receiving $\sigma_1, \ldots, \sigma_{\mathrm{T}}$. Notice also, that the weight function may be arbitrary, and in particular does not have to sum up to 1. As we work with bounded accuracy we assume $w(i,y)$ is always a multiple of $\frac{1}{\mathrm{M}}$. For normalization, we define the relative weight of w as

$$\rho(w) \quad = \quad \frac{\sum_{i \in [\mathrm{T}], y \in [\mathrm{M}]} w(i,y)}{\mathrm{TM}}.$$

Notice that $\rho(w)$ is a real number between zero and one.

We call the elements of $[\mathrm{T}] \times [\mathrm{M}]$ *points.* We view a word $u \in [\mathrm{M}]^{\mathrm{T}}$ as the set of points $\{(i,u_i) \ : \ i \in [\mathrm{T}]\}$. The *agreement* $\mathsf{Ag}(u,w)$ between a word u and a weight function w is the sum of the weights of the points corresponding to u, i.e., $\sum_{i \in [\mathrm{T}]} w(i,u_i)$. The case where w takes on only Boolean values is of special interest, as then w corresponds to sets. For sets $\mathcal{S}_1, \ldots, \mathcal{S}_{\mathrm{T}} \subseteq [\mathrm{M}]$, $\mathsf{Ag}(u,(\mathcal{S}_1, \ldots, \mathcal{S}_{\mathrm{T}})) = \mathsf{Ag}(u,w)$, where w is the indicator weight function $w(i,\sigma) = 1$ if $\sigma \in \mathcal{S}_i$ and 0 otherwise. When the sizes of the $\mathcal{S}_i$ are 1 this amounts to the usual notion of agreement between two vectors.

Now, let $\mathcal{C} = \{\mathcal{C}_r\}$ be a family of codes, where $\mathcal{C}_r \subseteq [\mathrm{M}]^{\mathrm{T}}$ is a code of length $\mathrm{T} = \mathrm{T}(r)$ over alphabet $[\mathrm{M}] = \{1,2,\ldots,\mathrm{M}\}$, $\mathrm{M} = \mathrm{M}(r)$. We use unconventional letters for describing the alphabet and the code length so as to avoid $n, N,\ k, K,$ and d, D which are standard for both extractors and coding theory. We say the family $\mathcal{C}$ has efficient encoding if there is a polynomial-time algorithm that given r and j computes the j'th codeword of $\mathcal{C}_r$ in time $\mathrm{poly}(\mathrm{M},\mathrm{T})$. When it's clear from the context we say that $\mathcal{C}$ has efficient encoding.

The list decoding problem comes in two flavors, combinatorial and algorithmic. In both we are given an arbitrary weight function w, which is the essence of the adversarial model. The combinatorial version is to bound the *number* of codewords that have large agreement with w, while the algorithmic version is to actually find all these codewords. Note that the expected agreement of a random word from $[\mathrm{M}]^{\mathrm{T}}$ with w is $\rho(w)\mathrm{T}$; by large agreement we mean noticeably larger than this, as is captured in the definition of $\mathcal{A}_\epsilon$ below.

Definition II.1: We say a code $\mathcal{C}_r \subseteq [\mathrm{M}]^{\mathrm{T}}$ has (L,ϵ) combinatorial soft decoding if for every weight function $w : [\mathrm{T}] \times [\mathrm{M}] \to [0,1]$ the set

$$\mathcal{A}_\epsilon(w) \quad = \quad \{u \in \mathcal{C} \ : \ \mathsf{Ag}(u,w) > (\rho(w)+\epsilon)\mathrm{T}\}$$

has size at most L. We say a family of codes $\mathcal{C} = \{\mathcal{C}_r\}$ has $(\mathrm{L} = \mathrm{L}(r), \epsilon = \epsilon(r))$ combinatorial soft decoding if for all r, $\mathcal{C}_r$ has $(\mathrm{L}(r),\epsilon(r))$ soft decoding. If $\mathrm{L}(r) \le p(\mathrm{T},\mathrm{M},\frac{1}{\epsilon})$ for some polynomial $p(\cdot)$ we simply say $\mathcal{C}$ has ϵ combinatorial soft decoding.

In the algorithmic version we want efficient decoding. That is, we have a noisy channel, and we see $\sigma_1, \ldots, \sigma_{\mathrm{T}}$ which induces a weight function $w : [\mathrm{T}] \times [\mathrm{M}] \to [0,1]$. We give the decoding algorithm black-box access to w, and we want to recover the original codeword.

Definition II.2: We say a family $\{\mathcal{C}_r\}$ has efficient ϵ soft decoding if there exists an algorithm that given black-box access to the weight function w outputs all the codewords in $\mathcal{A}_\epsilon(w)$ in time polynomial in $\mathrm{T},\mathrm{M},\frac{1}{\epsilon}$ [1].

We say a family $\{\mathcal{C}_r\}$ has efficient *probabilistic* ϵ soft decoding, if the decoding algorithm is probabilistic, and for every weight function w, with probability at least $1 - 2^{-(M+T)}$ over its internal random coins, outputs all the codewords in $\mathcal{A}_\epsilon(w)$ in time polynomial in $\mathrm{T},\mathrm{M},\frac{1}{\epsilon}$.

[1] Notice that we can represent a weight function $w : [\mathrm{T}] \times [\mathrm{M}] \to [0,1]$ with MT numbers, each a multiple of $\frac{1}{\mathrm{M}}$, and hence with $poly(\mathrm{T},\mathrm{M})$ bits. Thus, we could have replaced the above definition with one that says that given a description of w the algorithm outputs all codewords in $\mathcal{A}_\epsilon(w)$ in time $poly(\mathrm{T},\mathrm{M},\frac{1}{\epsilon})$. However, in many settings the black-box version is more natural.

We also define $\mathcal{A}_{\max}(w)$ to be the set of codewords $u \in \mathcal{C}$ that have full agreement with w, i.e., $\mathcal{A}_{\max}(w) = \{u \in \mathcal{C} \;:\; \mathsf{Ag}(u, w) = \mathrm{T}\}$.

B. Strong Extractors

As mentioned earlier, an extractor is a procedure to extract randomness from a defective random source, using a small additional number of truly random bits. In order to define extractors, we first give some standard definitions.

Definition II.3: A probability distribution D on Ω is a function $D : \Omega \rightarrow [0, 1]$ such that $\sum_{x \in \Omega} D(x) = 1$. For an integer n, U_n is the uniform distribution on $\{0, 1\}^n$. We overload notation and for a set S, U_S denotes the uniform distribution on S. The variation (statistical) distance between two probability distributions D_1 and D_2 on Ω, denoted $|D_1 - D_2|$, is $\frac{1}{2} \sum_{x \in \Omega} |D_1(x) - D_2(x)| = \max_{S \subseteq \Omega} |D_1(S) - D_2(S)|$. We say D_1 is ϵ close to D_2 if $|D_1 - D_2| \leq \epsilon$. A distribution on a set S is ϵ-uniform if it is ϵ–close to U_S.

We model a defective random source as one that has sufficient min-entropy, a notion that is more useful to us than entropy.

Definition II.4: The *min-entropy* of a distribution D is $min_x \log_2 1/D(x)$.

In other words, D has min-entropy ℓ if for all x, $D(x) \leq 2^{-\ell}$. We now define extractors, using slightly different parameters than usual (typically the L below is replaced by $\log \mathrm{L}$).

Definition II.5: A function $E : [\mathrm{C}] \times [\mathrm{T}] \rightarrow [\mathrm{M}]$ is a strong (L, ϵ)-*extractor* if for every distribution D on $[\mathrm{C}]$ such that for all x, $D(x) \leq 1/L$, the distribution $U_{\mathrm{T}} \circ E(D, U_{\mathrm{T}})$ obtained by picking x from D, y uniformly from $[\mathrm{T}]$ and evaluating $y \circ E(x, y)$ is ϵ-uniform. Here $\circ$ denotes concatenation. (E is an extractor if $E(D, U_{\mathrm{T}})$ is ϵ-uniform.)

In other words, for any test $W : [\mathrm{T}] \times [\mathrm{M}] \rightarrow \{0, 1\}$, the probability that $W(y, z) = 1$ is roughly the same whether y and z are chosen uniformly, or y is chosen uniformly and $z = E(x, y)$ for x chosen with sufficient min-entropy. Note that there is no constraint on the efficiency of the test; in fact, the test can be randomized, as a randomized test can be viewed as randomly picking one of several deterministic tests.

A natural interpretation for this is that $E : \{0, 1\}^c \times \{0, 1\}^t \rightarrow \{0, 1\}^m$ is a strong $(L = 2^\ell, \epsilon)$ extractor, if it takes an arbitrary distribution over $[C] = \{0, 1\}^c$ with at least ℓ min-entropy (this corresponds to $X \subseteq [C]$ of cardinality at least L), uses t independent truly random bits, and distills from X m output bits that are close to uniform (this corresponds to the fact that no test can distinguish $U_{\mathrm{T}} \circ E(D, U_{\mathrm{T}})$ from uniform). We would like, of course, to have t as small as possible and m as close as possible to ℓ.

We now define efficiency of extractors; first we give the usual notion and then a weaker version which will suffice for our purposes.

Definition II.6: An extractor $E : [\mathrm{C}] \times [\mathrm{T}] \rightarrow [\mathrm{M}]$ is efficient if it is computable in time polynomial in $\log \mathrm{T} + \log \mathrm{M} + \log \mathrm{C}$. It is weakly efficient if is computable in time polynomial in $\mathrm{T} + \log \mathrm{M} + \log \mathrm{C}$.

C. The Equivalence to Extractors

We associate a code with a strong extractor:

Definition II.7: Let $E : [\mathrm{C}] \times [\mathrm{T}] \rightarrow [\mathrm{M}]$ be a strong extractor. For each $c \in [\mathrm{C}]$, define a word $z(c) = (z_1, \ldots, z_{\mathrm{T}}) \in [\mathrm{M}]^{\mathrm{T}}$ where $z_i = E(c, i)$. The extractor code for E is $\mathcal{C}_E = \left\{ z(c) \in [\mathrm{M}]^{\mathrm{T}} \mid c \in [\mathrm{C}] \right\}$.

The next theorem shows that strong extractors give codes with good soft decoding, and that these two notions are, in fact, equivalent.

Theorem 1: If $E : [\mathrm{C}] \times [\mathrm{T}] \rightarrow [\mathrm{M}]$ is a strong (L, ϵ) extractor, then $\mathcal{C}_E$ has (L, ϵ) combinatorial soft decoding. Conversely, if $\mathcal{C}_E$ has (L, ϵ) combinatorial soft decoding, even if only with respect to Boolean weight functions, then E is a strong $(\frac{L}{\epsilon}, 2\epsilon)$ extractor. Finally, $\mathcal{C}_E$ has efficient encoding if and only if F is weakly efficient.

Note that this theorem also implies that combinatorial soft decoding with respect to Boolean weight functions implies general combinatorial soft decoding with only a slight degrading of parameters.

By efficient encoding, we mean that the encoding time is polynomial in the encoded length $\mathrm{T} \log \mathrm{M} \geq \log \mathrm{C}$. Therefore, the last statement in the theorem about efficiency is obvious, since encoding in $\mathcal{C}_F$ amounts to computing E at T different points.

Typically we will consider large alphabets: $\mathrm{M} \geq \mathrm{T}^{\Omega(1)}$ or larger. (An exception is when we consider the Johnson bound.) In this typical case, if we want the size L of the solution list to be polynomial in M (so that we have ϵ–combinatorial list decoding) the extractor E has to extract at least a constant fraction of the min-entropy in the given source. That is, if we denote $\mathrm{L} = 2^\ell$ (i.e., the given source X has ℓ min-entropy) and $\mathrm{M} = 2^m$ (i.e., the extractor extracts m bits that are close to uniform) then $L \leq \mathrm{M}^{O(1)}$, or equivalently $\mathrm{M} \geq \mathrm{L}^\alpha$ for some constant α, i.e., $m = \Omega(\ell)$.

Although a good extractor yields a code with good soft decoding and efficient encoding, usually we do not have efficient decoding for such codes. Getting codes with explicit encoding and decoding is a main goal of this paper, and we will later show how to achieve it.

The connection between extractors and error correcting codes sheds light on the Johnson bound [20], which states that a binary code with relative distance $\frac{1}{2} - \delta$ has the property that every ball of relative radius $\frac{1}{2} - \sqrt{\delta}$ contains at most $1/\delta$ codewords (the bound was generalized to larger fields in [25]). Typically, the Johnson bound is the key tool used to determine the quality of combinatorial list decoding in the traditional sense. The Johnson bound, however, does not generalize to the soft-decision setting (e.g., to the setting where all we know is that the i'th transmitted symbol is one of $\frac{|\Sigma|}{2}$ possibilities).

The Johnson bound was given several proofs; some of the more recent ones include a proof with geometric motivation in [26], a proof using inclusion-exclusion in [25] and a combinatorial proof using the Zarankiewicz bound in [27]. We, however, interpret the Johnson bound as saying that a good code is a good extractor, and we give it a simple proof that is identical to the simple proof of the Leftover

Hash Lemma [28]. In Lemma IV.1 we show that our interpretation does generalize to the soft-decision setting.

We also use the equivalence with extractors to derive bounds on soft decoding. A lower bound of [29] says that a strong (L, ϵ) extractor $E : [\mathrm{C}] \times [\mathrm{T}] \to [\mathrm{M}]$ must have

$$\mathrm{T} \quad \geq \quad \Omega(\frac{\log(\frac{C}{L})}{\epsilon^2}) \qquad (1)$$

and

$$L \quad \geq \quad \Omega(\frac{M}{T} \cdot \frac{1}{\epsilon^2}) \qquad (2)$$

Furthermore, random extractors, with high probability, almost match these bounds [29]. The equivalence of Theorem 1 translates this to non-explicit codes $\mathcal{C}_E \subseteq [\mathrm{M}]^{\mathrm{T}}$ with (L, ϵ) soft decoding and $|\mathrm{C}|$ codewords. The bound (1) then tells us how far can we push the rate of this code. Namely, the rate is $\frac{\log(\mathrm{C})}{\mathrm{T}\log(\mathrm{M})}$ which is about $\frac{\epsilon^2}{\log(\mathrm{M})}$ as long as L is much smaller than C. The bound (2) tells us that as long as the code length T is much smaller than the alphabet size M, the number of possible solutions cannot be constant. Anything is achievable within these bounds. In particular there are codes with many codewords and (L, ϵ) soft decoding for a relatively small $\mathrm{L} = \mathit{poly}(\mathrm{M}, \mathrm{T}, \frac{1}{\epsilon})$.

D. Reed-Solomon Codes

There has been a lot of research on efficiently encoding and decoding Reed-Solomon codes. We first state the beautiful list decoding result of Guruswami and Sudan [6] building upon Sudan [5].

Theorem 2: [6] Let $\mathcal{RS} \subseteq [\mathrm{M}]^{\mathrm{T}}$ be a Reed-Solomon code, $\mathrm{T} \leq \mathrm{M}$, $|\mathcal{RS}| = \mathrm{M}^c$. For every $\mathcal{S}_1, \ldots, \mathcal{S}_{\mathrm{T}} \subseteq [\mathrm{M}]$

$$|\Big\{u \in \mathcal{RS} \mid \mathsf{Ag}(u, (\mathcal{S}_1, \ldots, \mathcal{S}_{\mathrm{T}})) \geq \sqrt{c\mathcal{S}}\Big\}| \quad \leq \quad \mathrm{T}$$

where $\mathrm{S} = \sum_{i=1}^{\mathrm{T}} |\mathcal{S}_i|$. Furthermore, $\mathcal{RS}$ has efficient list decoding for these parameters.

Note that the bound is very good when S is small (say, about T) and that it deteriorates as S grows. In particular it is useless whenever $\sqrt{c\mathrm{S}} > \mathrm{T}$, (e.g., if $\mathrm{M} = \mathrm{T}$ and $S = \frac{\mathrm{MT}}{2}$). Restated in our soft decoding notation, the list decoding algorithm works well as long as $\rho(w)$ is small (about $\frac{1}{M}$), and is not guaranteed to work when $\rho(w)$ is large (a constant). We show that this phenomenon is not an artifact of the analysis but rather Reed-Solomon codes indeed have poor list decoding when $\rho(w)$ is large.

Theorem 3: Let $\mathcal{RS} \subseteq [\mathrm{M}]^{\mathrm{T}}$ be a Reed Solomon code with M^c elements, $\mathrm{T} \leq \mathrm{M}$, and M a prime power. Suppose $\mathrm{M} - 1 = b_1 b_2$ for some integers b_1 and b_2. Then, there is a Boolean weight function $w : [\mathrm{T}] \times [\mathrm{M}] \to \{0, 1\}$ with weight $\rho(w) = \frac{b_1+1}{M}$ for which $|\mathcal{A}_{\max}(w)| \geq \mathrm{M}^{\frac{c-1}{b_2}}$.

Thus, in particular, for $b_2 = 2$ we get a Boolean weight function of weight about half, for which there are about $\sqrt{|\mathcal{RS}|}$ close codewords. Also notice how the number of codewords in the bound decays with the relative weight of w.

E. An Extractor Code with Efficient Soft Decoding

We saw that good extractors E translate to codes $\mathcal{C}_E$ with good combinatorial soft decoding; in particular, if E extracts a constant fraction of the min-entropy of the given source (i.e., $m \geq \Omega(\ell)$) and has error ϵ, then $\mathcal{C}_E$ has ϵ *combinatorial* soft decoding. Nevertheless, not every good extractor suits our needs, as $\mathcal{C}_E$ might not have *efficient* soft decoding. Our main technical contribution is showing that block-box pseudorandom generator constructions translate to codes with efficient soft decoding. In Section VI we explain in detail the notion of block-box pseudorandom generator, and show that every code that originates from such a construction has efficient *probabilistic* soft decoding. In Section VII we show how to make the decoding algorithm deterministic for one such code.

Currently, there are two black-box pseudorandom generator constructions: Trevisan's construction [21] and Shaltiel and Umans's construction [30].

- Trevisan's extractor extracts half of the min-entropy in the source using $t = O(\log^2(\frac{c}{\epsilon}))$ truly random bits, hence its corresponding code $\mathcal{C}_{\mathrm{TR}} \subseteq [M]^T$ has ϵ combinatorial soft decoding. Its rate is $\frac{\log(C)}{T}\frac{1}{\log(M)} = \frac{1}{2^{(\log\log(\frac{C}{\epsilon}))^2}}\frac{1}{\log(M)}$, which should be compared to the best possible rate of about $\frac{\epsilon^2}{\log(M)}$. This reflects the fact that Trevisan's extractor has degree larger than optimal. Nevertheless, we still accommodate exponentially many codewords into $[\mathrm{M}]^{\mathrm{T}}$.
- Shaltiel and Umans's extractor extracts a sub-constant fraction of the min-entropy in the source, $m = \frac{\ell}{polylog(c)}$, using only $O(\log(\frac{c}{\epsilon}))$ truly random bits. Hence its corresponding code $\mathcal{C}_{\mathrm{SU}} \subseteq [M]^T$ has (L, ϵ) combinatorial soft decoding for a super-polynomial $L = M^{poly(\log\log(C))}$. Its rate is $\frac{\log(C)}{T}\frac{1}{\log(M)} = \frac{1}{2^{\log\log(\frac{C}{\epsilon})}}\frac{1}{\log(M)}$ which compares better to the lower bound which is about $\frac{\epsilon^2}{\log(M)}$.

Any future better black-box extractor construction would immediately translate to a better code. In particular, if such a construction is found with $m = \Omega(\ell)$ and $T = \Omega(\frac{\log(C)}{\epsilon^2})$ then we would get a code with efficient ϵ soft decoding and almost optimal rate.

One special case of interest is when $\mathrm{T} = \mathrm{M}$ which is similar to the Reed-Solomon case. It turns out that Trevisan's code has soft decoding for exponentially small ϵ. In fact, ϵ can be as small as $2^{-\Omega(\sqrt{\log \mathrm{M}})}$. On the other hand, Reed-Solomon codes do not have soft decoding for any $\epsilon > 0$. We again demonstrate this with the case where given a signal the receiver can only guess the transmitted symbol is one of $\frac{|\Sigma|}{2}$ possible symbols. The code $\mathcal{C}_{\mathrm{TR}}$ we presented can recover the original symbol even if just slightly over half the guesses are correct (half is what you expect from a random guess) whereas for Reed-Solomon codes there is no way to recover the original symbol even if every guess is always correct.

F. Applications

The first application of list decoding [4] was obtaining hardcore bits, though it was not observed to be a list decoding algorithm at the time. For this application, soft list decoding is more useful than ordinary list decoding. Using our codes we can output many hardcore bits while adding only few auxiliary random bits. In Section VIII we explain the problem, previous work and our result.

Recently, the results of this paper were used in [31] to give an explicit space-efficient method of storing a set of elements from a large universe in such a way that membership in the set can be determined (with high probability) by reading only one bit from the set representation. For the history of the problem, and the extensive previous work on it we refer the reader to [32].

III. Equivalence of Extractors and Soft Decoding

We now prove Theorem 1, the equivalence of extractors and codes with good soft decoding. The proof makes use of an alternate view of extractors suggested in [11], where for a given test the number of "bad" strings for this test is small.

Proof: [Of Theorem 1]:
E strong extractor $\Longrightarrow$ $\mathcal{C}_E$ has soft decoding :
Suppose $E : [\mathrm{C}] \times [\mathrm{T}] \to [\mathrm{M}]$ is a strong (L, ϵ) extractor. Let $w : [\mathrm{T}] \times [\mathrm{M}] \to [0,1]$ be a weight function for the code $\mathcal{C}_E \subseteq [\mathrm{M}]^{\mathrm{T}}$. Define the test $W : [\mathrm{T}] \times [\mathrm{M}] \to \{0,1\}$ which accepts (i,y) with probability $w(i,y)$. Then

$$\Pr[W(U_{[T]\times[M]})=1] = \frac{\sum_{i,y} w(i,y)}{\mathrm{TM}} = \rho(w)$$

We wish to show that the set $\mathcal{A} = \mathcal{A}_\epsilon(w)$ is small.
For a set $X \subseteq [\mathrm{C}]$, let $D_{E,X}$ be the distribution obtained by picking x uniformly from X, y uniformly from T and evaluating $y \circ E(x,y)$. By the definition of $\mathcal{A}$,

$$\Pr[W(D_{E,\mathcal{A}})=1] > \rho(w)+\epsilon.$$

Hence, $D_{E,\mathcal{A}}$ is not ϵ-uniform, so by the extractor definition we must have $|\mathcal{A}| \le \mathrm{L}$.

$\mathcal{C}_E$ has soft decoding $\Longrightarrow$ E is a strong extractor :
Fix a test $W : [\mathrm{T}] \times [\mathrm{M}] \to \{0,1\}$. As $\mathcal{C}_E$ has (L,ϵ) soft decoding, for at most L values of $x \in [\mathrm{C}]$,

$$\Pr[W(D_{E,\{x\}})=1] > \Pr[W(U_{[T]\times[M]})=1]+\epsilon.$$

Now let D be an arbitrary distribution on $[\mathrm{C}]$ with $D(x) \le \epsilon/\mathrm{L}$ for all x. Then

$$\Pr[W(U_{\mathrm{T}} \circ E(D,U_{\mathrm{T}}))=1] \quad \le \quad \Pr[W(U_{[T]\times[M]})=1]+2\epsilon,$$

since the L "bad" x only account for probability at most ϵ.

As this is true for every test W, and in particular for the test W' that negates W, we must have:

$$|\Pr[W(U_{\mathrm{T}} \circ E(D,U_{\mathrm{T}}))=1] - \Pr[W(U_{[T]\times[M]})=1]| \quad \le \quad 2\epsilon$$

Hence, E is a strong $(\frac{L}{\epsilon}, 2\epsilon)$ extractor. ■

IV. The Johnson bound

We start with the well known claim that codes with good distance properties are reasonable extractors:

Claim IV.1: (Codes are extractors) Let $\mathcal{C}$ be a $[\mathrm{T}, k, (1 - \frac{1}{q} - \delta)\mathrm{T}]_q$ code. Let $E : \mathbb{F}_q^k \times [\mathrm{T}] \to \mathbb{F}_q$ be the function $E(x,i) = \mathcal{C}(x)_i$. Then E is a $(\frac{1}{\delta}, \sqrt{\frac{q\delta}{2}})$ extractor.

Proof: We follow Rackoff's proof of the Leftover Hash Lemma as in [33]. Let $X \subseteq \mathbb{F}_q^k$ be of cardinality $\frac{1}{\delta}$, and recall that $D_{E,X}$ is the distribution obtained by picking y uniformly from [T], x uniformly from X, and evaluating $y \circ E(x,y)$.

The collision probability of $D_{E,X}$, denoted $col(D_{E,X})$, is the probability that two independent samples of the distribution $D_{E,X}$ are the same. It is therefore the probability over $y_1, y_2 \in [\mathrm{T}]$ and $x_1, x_2 \in X$ that $y_1 = y_2$ and $E(x_1,y_1) = E(x_2,y_2)$. Obviously we should have $y_1 = y_2$. If we also have $x_1 = x_2$ then we get a collision. From the large distance of the code $\mathcal{C}$ it follows that if $x_1 \ne x_2$ we have probability (over $y = y_1 = y_2$) of at most $\frac{1}{q} + \delta$ of having a collision. Altogether,

$$\begin{aligned} Col(D_{E,X}) &\le \frac{1}{\mathrm{T}}(\frac{1}{|X|} + \frac{1}{q} + \delta) \\ &= \frac{1}{q\mathrm{T}}(1 + \frac{q}{|X|} + q\delta) \end{aligned}$$

Rackoff shows that if D is distributed over some domain Ω and $col(D) \le \frac{1}{|\Omega|}(1+4\epsilon^2)$ then D is ϵ-uniform. It therefore follows that $D_{E,X}$ is ϵ-uniform where

$$\epsilon \quad \le \quad \frac{1}{2}\sqrt{\frac{q}{|X|} + q\delta} \quad = \quad \sqrt{q\delta/2} \qquad (3)$$

and E is a strong extractor. ■

The original Johnson bound [20] was for binary codes and was later generalized to arbitrary alphabets (e.g., in [25]). Our framework allows us to generalize the the second bound in [25] to the soft-decision setting.

Lemma IV.1: (Johnson bound - soft decoding setting) Let $\mathcal{C}$ be a $[\mathrm{T}, k, (1 - \frac{1}{q} - \delta)\mathrm{T}]_q$ code. Then for $\epsilon > \frac{1}{2}\sqrt{q\delta}$, $\mathcal{C}$ has $(\frac{q}{4\epsilon^2 - q\delta}, \epsilon)$ soft decoding.

Proof: Let $w : [\mathrm{T}] \times F_q \to [0,1]$ be a weight function. Let $\mathcal{A} = \mathcal{A}_\epsilon(w)$ and F, $D_{F,\mathcal{A}}$ be as before. On the one hand, for every $u \in \mathcal{A}$ we have $\mathsf{Ag}(u,w) > (\rho(w)+\epsilon)\mathrm{T}$ and therefore $|D_{F,\mathcal{A}} - U_{[T]\times[M]}| > \epsilon$. On the other hand, from Equation (3) it follows that $|D_{F,\mathcal{A}} - U_{[T]\times[M]}| \le \frac{1}{2}\sqrt{\frac{q}{|\mathcal{A}|} + q\delta}$. Putting it together we see that $|\mathcal{A}| < \frac{q}{4\epsilon^2 - q\delta}$. ■

A special case of the theorem is when $q = 2$ and we look at regular list decoding. We then get that if the relative distance is $\frac{1}{2} - \delta$ then any ball of relative radius $\frac{1}{2} - \sqrt{\delta}$ contains at most $\frac{1}{\delta}$ code words, which is the original Johnson bound.

V. Reed-Solomon Codes

Proof: (of Theorem 3) The proof uses the multiplicative subgroups of $\mathbb{F}_{\mathrm{M}}^*$, an idea that was exploited in a more sophisticated way in [34].

For $b_1|\mathrm{M}-1$, $\mathbb{F}_{\mathrm{M}}^*$ contains b_1 distinct solutions to the equation $x^{b_1}=1$. Denote these solutions by $\{\alpha_1,\ldots,\alpha_{b_1}\}$, and define $\alpha_0=0$. Let $\mathcal{S}=\{(i,\alpha_j)\ :\ i\in[\mathrm{T}], 0\le j\le b_1\}$, $|\mathcal{S}|=\mathrm{T}(b_1+1)$ and define the Boolean weight function $w:[\mathrm{T}]\times[\mathrm{M}]\to\{0,1\}$ to be one on $\mathcal{S}$ and zero otherwise.

Let $\mathcal{A}$ be the set of polynomials p^{b_2} for some p of degree at most $\frac{c-1}{b_2}$. For every $q=p^{b_2}\in\mathcal{A}$ and i, either $p(i)=0$ and then $q(i)=0$ or $p(i)\neq 0$ and then $(q(i))^{b_1}=(p(i))^{b_1b_2}=(p(i))^{\mathrm{M}-1}=1$, and so $(i,q(i))\in\mathcal{S}$. Hence, if $\mathcal{RS}(q)$ denotes the codeword corresponding to q, $\mathsf{Ag}(\mathcal{RS}(q),\mathcal{S})=\mathrm{T}$. Since distinct q of degree less than c give distinct codewords $\mathcal{RS}(q)$, $\mathcal{A}_{max}(w)\ge|\mathcal{A}|$, where $\mathcal{A}_{max}(w)$ is the set of all codewords u that have full agreement with w, i.e., $\mathsf{Ag}(u,w)=T$.

To lower bound $|\mathcal{A}|$, note that $\mathbb{F}_{\mathrm{M}}[X]$ is a unique factorization domain, which implies that for each $q\in\mathcal{A}$, the p such that $p^{b_2}=q$ is uniquely determined up to multiples of $\mathbb{F}_{\mathrm{M}}$. Consequently, each $q\in\mathcal{A}$ can arise from at most M polynomials p of degree at most $\frac{c-1}{b_2}$ (in fact, at most b_2 such polynomials). Therefore $|\mathcal{A}|\ge \mathrm{M}^{1+\frac{c-1}{b_2}}/\mathrm{M}$.

■

VI. Extractors and Pseudorandom Generators

In this section, we show that extractors constructed via so-called black-box pseudorandom generators give extractor codes with efficient probabilistic soft decoding. In the next section, we show how to make the soft decoding deterministic for one such extractor, Trevisan's extractor [21]. The next section does not rely on this section, so some readers who wish to avoid more computer science terminology may skip directly to that section. However, we believe that this section is more basic. We begin with the necessary background.

A. Background on Non-uniformity and Circuits

A non-uniform algorithm A is an infinite sequence of algorithms $A_1,A_2,\ldots$, one for each input size. To run A on an input x of size n, we run A_n on x. As one might expect, an infinite sequence of algorithms is more powerful than one algorithm; intuitively, it may be hard to compute the algorithm A_n from n.

Traditionally, the A_i are modeled as circuits; here it is natural to have a different circuit for each input size. An equivalent way to model A is as a Turing machine with advice. On an input x of size n, A receives some advice string a_n depending only on n; A then performs its computation on x and a_n. One may think of a_n as a description of the circuit.

For more background in this area, we refer the reader to textbooks in computational complexity, such as [35].

B. Pseudorandom Generators

A pseudorandom generator takes a short random string and expands it to a long string that looks random to all small circuits.

Definition VI.1: Let D_1,D_2 be two distributions on Ω. We say a circuit A ϵ–distinguishes D_1 from D_2 if

$$|\Pr_{x_1\in D_1}[A(x_1)=1]-\Pr_{x_2\in D_2}[A(g_2)=1]\ |\quad\ge\quad\epsilon$$

If D_2 is uniform on Ω we say A ϵ–distinguishes D_1 from uniform.

Definition VI.2: $G\ :\ [\mathrm{T}]\ \to\ [\mathrm{M}]$ is a strong ϵ–pseudorandom generator against size s circuits, if no size s circuit ϵ–distinguishes the distribution $U_{[T]}\circ G(U_{[T]})$ from uniform, where the distribution $U_{[T]}\circ G(U_{[T]})$ is obtained by picking y uniformly from [T] and evaluating $y\circ G(y)$. (A pseudorandom generator without the word strong would only require that no small circuit distinguishes $G(U_{[T]})$ from uniform.)

Note that for strong pseudorandom generators, we require the circuit size s to be smaller than the time to compute G; otherwise, the circuit could compute G and distinguish the output from uniform.

Currently, no explicit pseudorandom generators are known. However, it is known how to build pseudorandom generators from a given hard function f. I.e., there is a general construction $G=G_f:[\mathrm{T}]\to[\mathrm{M}]$, that is guaranteed to be pseudorandom for small circuits whenever f is hard for small circuits. Such a construction first appeared in [36] and was later improved in [37], [38], [12]. A second construction appears in [30]. In these constructions, correctness is proved by proving the contrapositive. If G_f is *not* a pseudorandom generator, then by definition there exists a small circuit A distinguishing $U_{[T]}\circ G_f(U_{[T]})$ from uniform. The proof then shows how to use the distinguisher A to build a small circuit for f. As f is hard for small circuits, one must conclude that G_f is pseudorandom.

A notable property of all current pseudorandom constructions is that they are *block-box.* This means that the generator $G=G_f$ only needs black-box access to f, i.e., it only needs f values on given inputs, and otherwise does not use any other operty of f. Similarly, the reconstruction algorithm that computes f using the distinguisher A, only needs black-box access to evaluations of A, and otherwise does not use any other property of A. This is captured in the definition of a black-box pseudorandom generator below. Note that an *oracle* Turing machine (circuit) is a Turing machine (circuit) that has black-box access (also called oracle access) to a function.

Definition VI.3: A black-box generator is an oracle machine $G^f:[T]\to[M]$, with black-box access to a Boolean function $f:[\log(\mathrm{C})]\to\{0,1\}$. We say G is efficient if it runs in time polynomial in its input length $\log(T)$.

Definition VI.4: A black-box reconstruction algorithm for G^f with a bits of advice, is an algorithm R that takes a short advice string $adv=adv(f)\in\{0,1\}^a$ and outputs an oracle circuit $R(a)^A$. Let $Time(R)$ be the running time of R and $Size(B)$ the size of the oracle circuit implementing B. R may be uniform or non-uniform, deterministic or probabilistic.

Definition VI.5: We say (G^f,R) is a black-box ϵ–pseudorandom generator if for every Boolean function $f:[\log(C)]\to\{0,1\}$, and every circuit A ϵ–distinguishing

$U_{[T]} \circ G^f(U_{[T]})$ from uniform, there exists an advice string $adv = adv(f) \in \{0,1\}^a$ such that $R(adv)^A = f$. If R is probabilistic, this means that with high probability over the random coins of R, the output $B = R(adv)^A$ is an oracle circuit, that with oracle access to A correctly computes f.

Note that the definition of a black-box pseudorandom generator does not refer to the size of circuits. However, when we instantiate f to be a hard function for certain size circuits, we do obtain a pseudorandom generator for small circuits.

We now claim:

Proposition VI.1: ([36]) Let (G^f, R) be a black-box ϵ–pseudorandom generator with $B = R(adv)^A$ the oracle circuit output by R. If f can not be computed by size $s \cdot Size(B)$ circuits, then G^f is a strong ϵ–pseudorandom generator for size s circuits.

Proof: Let $f : [\log(C)] \to \{0,1\}$ be an arbitrary Boolean function. If there exists a size s circuit A ϵ–distinguishing the distribution $U_{[T]} \circ G^f(U_{[T]})$ from uniform, then given the right advice $adv = adv(f)$, $B^A = R(adv)^A$ computes f. We can then take the oracle circuit for B and replace each oracle call to A with a size s circuit. We thus get a size $size(B) \cdot s$ circuit evaluating f, contradicting the hardness of f. ■

Notice that for the proposition to be useful, $Size(B)$ has to be small. On the other hand, the argument does not require that the reconstruction algorithm R be uniform or efficient.

C. A Probabilistic Soft Decoding Algorithm

Trevisan showed that a black-box pseudorandom generator G^f gives rise to a strong extractor $E : [C] \times [T] \to [M]$ defined by $E(f, y) = G^f(y)$ [21]. Here $f \in$ [C] is identified with a function $f : [\log(C)] \to \{0,1\}$. We rephrase his proof to say that the corresponding extractor code $\mathcal{C}_E$, defined by $\mathcal{C}_E(f) = (G^f(y_1), \ldots, G^f(y_T))$, has good soft decoding. Further, we extend it to say that if the reconstruction procedure is efficient (and hence the output circuit has small size $Size(B)$), then the code has efficient soft decoding.

Proposition VI.2: Suppose (G^f, R) is a black-box pseudorandom generator with R using a bits of advice. Then the extractor code $\mathcal{C}_E$ for $E(f, y) = G^f(y)$ has $(L = 2^a, \epsilon)$-combinatorial soft decoding with respect to Boolean weight functions. If R is efficient, then so is this soft decoding.

It follows from Theorem 1 that E is a strong $(2^a/\epsilon, 2\epsilon)$ extractor and $\mathcal{C}_E$ has $(2^a/\epsilon, 2\epsilon)$-combinatorial soft decoding.

Proof: Let $w : [T] \times [M] \to \{0,1\}$ be a Boolean weight function, and let W be a circuit computing w. Let $\mathcal{A}$ be the set of functions corresponding to $\mathcal{A}_\epsilon(w)$, i.e., let $\mathcal{A}$ be the set of all functions $f \in$ C with $\mathsf{Ag}(\mathcal{C}(f), w) > (\rho(w) + \epsilon)T$. Then for any $f \in \mathcal{A}$, W ϵ-distinguishes $U_{[T]} \circ G^f(U_{[T]})$. This is because $\Pr[W(U_{[T]}, U_{[M]}) = 1] = \rho(w)$ and

$$\Pr_{y \in U_{[T]}}[W(y, E(f, y)) = 1] = \sum_y \frac{1}{T} w(y, E(f, y)) = \frac{\mathsf{Ag}(\mathcal{C}(f), w)}{T}$$

Therefore, the reconstruction property implies that for every $f \in \mathcal{A}$ there exists an advice string $adv = adv(f)$ such that $R(adv(f))^W = f$. In particular, different functions $f \in \mathcal{A}$ must have different advice strings $adv(f)$. We conclude that the number of functions in $\mathcal{A}$ is at most 2^a, as required.

If R is efficient, then the decoding algorithm is to cycle over all 2^a advice strings adv. For each of the 2^a functions $f = R(adv)^W$, test if $f \in \mathcal{A}$, and if so, output $\mathcal{C}(f)$. By the previous argument, this outputs every codeword $\mathcal{C}(f)$ with $f \in \mathcal{A}$. The running time is polynomial in L and the running time of R (since the size of $R(adv)$ is at most the running time of R). ■

We now extend the previous proposition to soft decoding with arbitrary weight functions:

Proposition VI.3: Suppose (G^f, R) is a black-box pseudorandom generator with R using a bits of advice, where R is efficient. Then the extractor code $\mathcal{C}_E$ for $E(f, y) = G^f(y)$ has $(L = \frac{2^a}{\epsilon}, 2\epsilon)$ combinatorial soft decoding and efficient probabilistic 2ϵ–soft decoding.

Proof: As mentioned earlier, the combinatorial soft decoding follows from Proposition VI.2 and Theorem 1. We now describe the efficient decoding.

Let $w : [T] \times [M] \to [0,1]$ be a weight function. Define the probabilistic test $W : [T] \times [M] \to \{0,1\}$ that accepts $(y, z) \in [T] \times [M]$ with probability $w(y, u)$. As before, let $\mathcal{A}$ be the functions corresponding to $\mathcal{A}_{2\epsilon}(w)$. Then, for the same reason as in Proposition VI.2, for any $f \in \mathcal{A}$, W 2ϵ-distinguishes $U_{[T]} \circ G^f(U_{[T]})$. We are therefore in the same situation as in Proposition VI.2, but now we have a probabilistic distinguisher W. However, the property of the black-box pseudorandom generator only applies to deterministic circuits.

We get around this by viewing the probabilistic test W as a probability distribution over deterministic tests $\{W_r\}$, where r are the internal coins of W. As W 2ϵ–distinguishes $U_{[T]} \circ G^f(U_{[T]})$ from uniform, it must be that for at least ϵ faction of the random coins r, W_r ϵ–distinguishes $U_{[T]} \circ G^f(U_{[T]})$ from uniform. Let us call such an r good for f. If r is good for f, then the soft decoding algorithm from Proposition VI.2 running on W_r will output f.

We therefore define our decoding algorithm to toss $s = 2a/\epsilon^2$ random strings $r_1, \ldots, r_s$. For each r in the sequence, run the decoding algorithm from Proposition VI.2 on W_r. For fixed f, the probability that none of $r_1, \ldots, r_s$ are good for f is at most $(1 - \epsilon)^s \le \exp(-2a/\epsilon)$. Thus, this bounds the probability that f is not output. By the union bound, the probability there exists some $f \in \mathcal{A}$ that is not output is at most $L \exp(-2a/\epsilon) \le 2^{-a}$.

■

VII. A Deterministic Soft Decoding Algorithm

Proposition VI.3 gives us a *probabilistic* soft decoding algorithm. Our next goal is to find a *deterministic* soft decoding algorithm. Looking back at the decoding algorithm in Proposition VI.3 we see that the problem is it uses a probabilistic distinguisher $A = \{A_r\}$. While many choices for r work, some do not and the decoding algorithm then may (with very low probability, yet non-zero) sample $r_1, \ldots, r_s$ that are all bad.

To rectify this problem we observe that current pseudorandom black-box generators are designed very much like concatenated codes. We then show how to reduce soft decoding of the concatenated code to soft decoding of the inner code. We take the inner code to be Reed-Solomon concatenated with Hadamard, and we show an explicit soft decoding for it. Together, we get a deterministic soft decoding algorithm.

The above approach works for both Trevisan's extractor and Shaltiel and Umans' extractor. We work out the details for Trevisan's extractor [21] because technically it is simpler.

A. Weak Designs

We first present the extractor [21] following the somewhat improved version given in [39]. We begin with some definitions.

Definition VII.1 (Weak design) [39] A family of sets $Z_1, Z_2, \ldots, Z_m \subseteq [t]$ is a weak (s, ρ) design if
1. $\forall i\ |Z_i| = s$, and
2. $\forall i,\ \sum_{j<i} 2^{|Z_i \cap Z_j|} \le \rho \cdot (m-1)$.

We have:

Lemma VII.1: [39] For every s, m and $\rho > 1$, there exists a weak (s, ρ) design $Z_1, Z_2, \ldots Z_m \subseteq [t]$ with $t = \left\lceil \frac{s}{\ln \rho} \right\rceil \cdot s$. Such a family can be found in time $\mathrm{poly}(m, t)$.

We introduce some notation for specifying substrings indexed by weak designs. For $y = y_1 \ldots y_t \in \{0,1\}^t$ and $Z = \{z_1, \ldots, z_s\} \subseteq [t]$ we denote by $y|Z$ the nonnegative integer obtained by restricting y only to those entries that are in Z, i.e. $j = y|Z$ is the number whose binary representation is $y_{z_1} y_{z_2} \ldots y_{z_s}$.

Also, for strings $\beta \in \{0,1\}^{t-s}$ and $\gamma \in \{0,1\}^s$ we denote by $\gamma^Z \circ \beta^{[t] \setminus Z}$ the t bit long string that has γ in the locations indexed by Z and β in the other locations.

B. Soft Decoding Reed-Solomon concatenated with Hadamard

We will need to use a soft decodable binary code; we choose to use Reed-Solomon concatenated with Hadamard. Let $\mathcal{RS}$ be an $[n, k, n-k+1]_n$ Reed-Solomon code. Let H be an $[n, \log n, \frac{n}{2}]_2$ Hadamard code. The concatenated code $\mathcal{BC}$ is an $[\bar{n} = n^2, k \log n, \frac{n(n-k+1)}{2}]_2$ code. A simple calculation shows that if we introduce a new parameter δ than $\mathcal{BC}$ is an $[\bar{n}, k, (\frac{1}{2} - \delta)\bar{n}]_2$ code with $\delta = \frac{k}{2n}$ (and $\bar{n} = O((\frac{k}{\delta})^2)$). The Johnson bound shows that $\mathcal{C}$ has good list decoding, and its generalization shows that $\mathcal{C}$ has good soft decoding.

Explicit list decoding is also not hard. In [40] improved explicit list decoding is shown. A soft decoding algorithm can be obtained in a standard way by first soft decoding the inner Hadamard code with brute force (the number of Hadamard codewords is only n so we can cycle over all of them) and then using list-decoding for the outer Reed-Solomon code. Details follow:

Lemma VII.2: Let $\mathcal{C} \subseteq [\mathrm{M}]^{\mathrm{T}}$ be the $[\bar{n}, k, (\frac{1}{2} - \delta)\bar{n}]_2$ Reed-Solomon concatenated with Hadamard code as above, $\mathrm{M} = 2$, $\mathrm{T} = \bar{n}$. Then $\mathcal{C}$ has $4\delta^{1/4}$ efficient soft decoding.

Proof: We first define the soft decoding algorithm, having black-box access to some weight function $w : [\mathrm{T}] \times [\mathrm{M}] \to [0,1]$. We view the output bits as composed of $\frac{\bar{n}}{n} = n$ blocks of length n each. For each block j, the weight function w induces a weight function $w_j : [n] \times \{0,1\} \to [0,1]$ on the block, where $w_j(i, \sigma)$ is the weight w gives to σ in the i'th location of the the j'th block.

For each block j, we run a $2\delta^{1/4}$ Hadamard soft decoding by brute force (i.e., by enumerating all n codewords in the Hadamard code, and checking the weights) and get a list S_j of all Hadamard codewords in $\mathcal{A}_{2\delta^{1/4}}(w_j)$. By Lemma IV.1, $|S_j| \le \frac{1}{8\sqrt{\delta}}$. We now apply Theorem 2 and find all Reed-Solomon codewords with $2\delta^{1/4}n$ agreement with the sets $S_1, \ldots, S_n$.

Every codeword of $\mathcal{C}$ that is also in $\mathcal{A}_{4\delta^{1/4}}(w)$, must have at least $2\delta^{1/4}n$ blocks j that belong to $\mathcal{A}_{2\delta^{1/4}}(w_j)$, and therefore must appear in the soft-decoding output list. Also, the parameters were chosen such that $2\delta^{1/4} \ge \sqrt{k|S|}$ (keeping in mind that $\delta = \frac{k}{2n}$), and so by Theorem 2 the procedure outputs at most n codewords of $\mathcal{C}$, and the whole procedure is efficient. ■

C. Trevisan's Extractor

Trevisan [21] constructed an extractor $\mathrm{TR_{c,m,\epsilon}(x, y)}$ as follows. Compute a weak (s, ρ) design $Z_1, Z_2, \ldots Z_m \subseteq [t]$ as in Lemma VII.1. Encode the input $x \in \{0,1\}^c$ using the binary error correcting code $\mathcal{BC}$ given in Lemma VII.2 to get a $\bar{c} = poly(c, \frac{1}{\beta})$ long string $\hat{x} = \mathcal{BC}(x)$. Use the random string y to select m output bits from $\hat{x}$. The i'th output bit is $\hat{x}_j$, where $j = y|Z_i$.

Trevisan's extractor [21] $\mathrm{TR_{c,m,\epsilon}}$.

Parameters : $c, m, \epsilon > 0$. $\rho = 2$.
Binary code : $\mathcal{BC}_{c,\delta} : \{0,1\}^c \to \{0,1\}^{\bar{c}}$ the binary code given by Lemma VII.2 with $\delta = (\frac{\epsilon}{4m})^4$.
Weak Design : A weak (s, ρ) design $Z_1, \dots, Z_m \subseteq [t]$, with:

- $s = \log \bar{c} = O(\log c + \log m + \log(\frac{1}{\epsilon}))$,
- $t = s\lceil \frac{s}{\ln \rho} \rceil = O((\log c + \log m + \log(\frac{1}{\epsilon}))^2)$

Input : $x \in \{0,1\}^c$.
Random string : $y \in \{0,1\}^t$.
Output : The output has m bits,

$$\mathrm{TR(x;y)_i = \hat{x}(y|Z_i)}$$

where $\hat{x} = \mathcal{BC}(x)$.

Letting $\mathrm{T} = 2^t$ and $\mathrm{M} = 2^m$, note that $\mathcal{C}_{\mathrm{TR}} \subseteq [\mathrm{M}]^{\mathrm{T}}$ is of size 2^c.

D. Decoding $\mathcal{C}_{\mathrm{TR}}$

We now describe the decoding algorithm for $\mathcal{C}_{\mathrm{TR}}$.

Algorithm VII-D: Decoding $\mathcal{C}_{\mathrm{TR}}$.

Input : $w : [\mathrm{T}] \times [\mathrm{M}] \to [0,1]$.

Algorithm : For every:

- $\alpha \in \{0,1\}^m$,
- $\beta \in \{0,1\}^{t-s}$,
- $1 \le i \le m$,
- set of possible truth tables $P_1, \dots, P_{i-1}$, where P_j has size $2^{|Z_i \cap Z_j|}$,

define a weight function $w_{\mathcal{BC}}$ for the binary code $\mathcal{BC}$ as follows. For every $\gamma \in \{0,1\}^s$ and $bit \in \{0,1\}$:

- Let $y = \gamma^{Z_i} \circ \beta^{[t] \setminus Z_i}$.
- Define $b \in \{0,1\}^m$ by:

$$b_j \quad = \quad \begin{cases} P_j(y|_{Z_i \cap Z_j}) & \text{if } j < i \\ bit & \text{if } j = i \\ \alpha_j & \text{if } j > i \end{cases}$$

and we let $w_{\mathcal{BC}}(\gamma, bit) = w(y, b)$.

Find all codewords $\hat{v} \in \mathcal{BC}$ with $\mathsf{Ag}(\hat{v}, w_{\mathcal{BC}}) \ge (\frac{1}{2} + \frac{\delta}{m})\bar{c}$ and output the corresponding $v \in \{0,1\}^c$.

Lemma VII.3: For every $w : \{0,1\}^t \times \{0,1\}^m \to [0,1]$ and every $x \in \{0,1\}^c$ for which $\mathsf{Ag}(x, w) \ge (\rho(w) + \epsilon)\mathrm{T}$, x appears in the output list of Algorithm VII-D.

The proof follows the outline in Section VI, and is a "constructive" variation of Trevisan's proof, giving an efficient algorithm "decoding" the extractor. A technical difference between this proof and Trevisan's proof, is that our decoding algorithm reduces to a soft decoding algorithm of the binary code $\mathcal{BC}$.

Proof: Fix any $x \in \{0,1\}^c$ such that $\mathsf{Ag}(x, w) \ge (\rho(w) + \epsilon)\mathrm{T}$. Define the probabilistic test

$$W : \{0,1\}^{t+m} \to \{0,1\}$$

which accepts (y, u) with probability $w(y, u)$. As before, let $D_{\mathrm{TR,\{x\}}}$ be the distribution $U_{[T]} \circ \mathrm{TR(\{x\}, U_{[T]})}$ obtained by picking y uniformly at random from $\{0,1\}^t$ and computing $y \circ \mathrm{TR(x,y)}$.

We now define the reconstruction procedure R. Define the hybrid distributions $D_0, \dots, D_m$ where D_i picks the first $t + i$ bits from $D_{\mathrm{TR,\{x\}}}$ and the last $m - i$ bits from U_{t+m}. We have

$$\Pr[W(D_0) = 1] \;=\; \Pr[W(U_{t+m}) = 1] \;=\; \rho(w)$$

and

$$\Pr[W(D_m) = 1] \;=\; \Pr[W(D_{\mathrm{TR,x}}) = 1] \;\ge\; \rho(w) + \epsilon$$

Therefore there must be an i, $1 \le i \le m$, such that

$$\Pr[W(D_i) = 1] \quad \ge \quad \Pr[W(D_{i-1}) = 1] + \frac{\epsilon}{m}$$

In both D_{i-1} and D_i, the last $m - i$ bits are uniform and independent of the rest. Hence there exists a fixed string $\alpha \in \{0,1\}^m$ such that fixing the last $m - i$ bits to the last bits of α preserves the gap of at least ϵ/m.

We can also split $y \in \{0,1\}^t$ into those bits in locations indexed by Z_i, which we denote by γ, and the rest of the bits in locations $[t] \setminus Z_i$ which we denote by β (and so $y = \gamma^{Z_i}\beta^{[t]\setminus Z_i}$). Again, there is a way to fix β and still preserve the gap.

Furthermore, once β is fixed the values $\mathrm{TR(x,j)}$ for $j = 1, \dots, i - 1$, that appear on both distributions, depend only on the bits of γ and in a very weak way. In particular there are truth tables $P_1, \dots, P_{i-1}$ that describe these dependencies (the string describing the truth tables $P_1, \dots, P_{i-1}$ corresponds to the advice string $adv = adv(x)$ in Section VI).

Now, for the given x we have:

$$\Pr_{\gamma}[W(y \circ P_1(\gamma) \dots P_{i-1}(\gamma)\ \hat{x}(\gamma)\ \alpha_{i+1} \dots \alpha_m) = 1] \;\ge$$

$$\Pr_{b_i \in \{0,1\}, \gamma}[W(y \circ P_1(\gamma) \dots P_{i-1}(\gamma)\ b_i\ \alpha_{i+1} \dots \alpha_m) = 1] + \frac{\epsilon}{m}$$

Now notice that $\mathbf{E}_{\gamma \in \{0,1\}^s}\ [w_{\mathcal{BC}}(\gamma, \hat{x}_\gamma)]$ is

$$\Pr_{\gamma}[W(y \circ P_1(\gamma) \dots P_{i-1}(\gamma)\ \hat{x}_\gamma\ \alpha_{i+1} \dots .) = 1]$$

and $\mathbf{E}_{bit \in \{0,1\}, \gamma}\ [w_{\mathcal{BC}}(\gamma, bit)]$ is

$$\Pr_{b_i \in \{0,1\}, \gamma}[W(y \circ P_1(\gamma) \dots P_{i-1}(\gamma)\ b_i\ \alpha_{i+1} \dots .) = 1]$$

We therefore conclude that

$$\mathbf{E}_{\gamma}\ [w_{\mathcal{BC}}(\gamma,\hat{x}_{\gamma})] \quad\geq\quad \mathbf{E}_{bit\in\{0,1\},\gamma}\ [w_{\mathcal{BC}}(\gamma,bit)]+\frac{\epsilon}{m}$$

which means that

$$\frac{\mathsf{Ag}(\hat{x},w_{\mathcal{BC}})}{\bar{c}} \quad\geq\quad \rho(w_{\mathcal{BC}})+\frac{\epsilon}{m}$$

Since $\frac{\epsilon}{m}\geq 4\delta^{1/4}$ Lemma VII.2 asserts that x appears in the output. ■

The number of elements in the output list is at most $2^m\cdot 2^t\cdot m\cdot 2^{\rho m}\cdot O(\frac{m}{\epsilon})^4$ because we have 2^m possibilities for α, 2^t possibilities for β, m possibilities for i and $2^{\rho m}$ possibilities for the tables $P_1,\ldots,P_{i-1}$, and then we get at most $\frac{1}{\delta}=O((\frac{m}{\epsilon})^4)$ possible answers (see Lemma VII.2). Taking $\rho=2$, $c\leq \mathrm{M}$ we get a running time of $\mathrm{poly}(\mathrm{M},\mathrm{T},\frac{1}{\epsilon})$, which completes the proof of Theorem.

VIII. Hardcore bits

One-way functions and their hardcore bits are key tools in cryptography. Informally, a one-way function f is easy to compute but hard to invert. By inverting z we mean finding any element of $f^{-1}(z)$. A function h on the same domain as f is hardcore for f if all statistical information about $h(x)$ is hard to predict even given $f(x)$. We usually allow h to also have access to auxiliary randomness. We formalize all this as follows.

Definition VIII.1: $f:[C]\rightarrow S$ is a one-way function with security (K,ϵ) if f is computable in time $\sqrt{K}$ but no probabilistic algorithm running in time K can invert f with probability at least ϵ, where the probability is over a random input and the coins of the inverting algorithm.

Typically, the time to invert f is conjectured to be superpolynomial in the time to compute f, but we only need the above small difference for our result.

Definition VIII.2: Let $f:[C]\rightarrow S$ be a function. $h:[C]\times[T]\rightarrow[M]$ is a hardcore function for f, with security (K,ϵ), if there is no probabilistic algorithm running in time K that ϵ–distinguishes[2] the distributions $(f(C),U_{[T]},h(C,U_{[T]}))$ and $(f(C),U_{[T]},U_{[M]})$, where $(f(C),U_{[T]},h(C,U_{[T]}))$ is the distribution obtained by picking c uniformly from C, y uniformly from $[T]$ and evaluating $(f(c),y,h(c,y))$, and $(f(C),U_{[T]},U_{[M]})$ is the distribution obtained by picking c uniformly from C, y uniformly from $[T]$, z uniformly from $[M]$ and evaluating $(f(c),y,z)$.

We say h is *hardcore* with security (K,ϵ) if it is hardcore with security (K,ϵ) for all one-way functions with security $(K^4,\frac{\epsilon}{4})$.

Goldreich and Levin [4] showed that for $T=C=\{0,1\}^c$, the function $h(x,y)=\oplus_i x_i y_i$ is a hardcore bit (with $M=\{0,1\}^{m=1}$). Note that this requires as many auxiliary random bits y as input bits x, and that the hardcore function has only one output bit.

Impagliazzo [41] noticed that this is a special case of a generic construction based on list decodable codes. Let $\mathcal{C}:[C]\rightarrow\{0,1\}^{\mathrm{T}}$ be a list decodable binary code. Then $h(x,y)=\mathcal{C}(x)_y$ (the yth bit of the encoding of x) is a hardcore predicate. Thus, the Goldreich and Levin result can be viewed as a list decoding algorithm for the Hadamard code. Moreover, by using list decodable binary codes with better rate, such as Reed-Solomon concatenated with Hadamard, a hardcore bit can be obtained with few auxiliary random bits. The hardcore function, though, still has only one output bit.

The main point of this section is to obtain many hardcore bits using small auxiliary randomness. This follows from our observation that Impagliazzo's construction generalizes to the case of more output bits. To get more bits, we choose a code over a larger alphabet, $\mathcal{C}:[C]\rightarrow[\mathrm{M}]^{\mathrm{T}}$, and the hardcore bits are $h(x,y)=\mathcal{C}(x)_y$. The notion of soft list decoding turns out to be what is needed here.

Theorem 4: Let $\mathcal{C}:[C]\rightarrow[\mathrm{M}]^{\mathrm{T}}$ be a code with a $\frac{\epsilon}{4}$ soft decoding algorithm, running in time K, $\epsilon\geq 1/K$. Then $h(x,y)=\mathcal{C}(x)_y$ is hardcore with security (K,ϵ).

Proof: Suppose there exists a one-way function f with security $(K^4,\epsilon/4)$ and a statistical (possibly probabilistic) test D which takes time K and ϵ-distinguishes the distributions $(f(C),U_{[T]},h(C,U_{[T]}))$ and $(f(C),U_{[T]},U_{[M]})$. We now give an inversion algorithm for f. The inversion algorithm is given an input $z\in S$ to invert. We first define a weight function $w=w_z:[T]\times[M]\rightarrow[0,1]$ by letting $w(y,v)$ be the probability that D accepts the input (z,y,v). As D may be probabilistic, we do not have direct access to w. Nevertheless, we can estimate $w(y,v)$ by random sampling, and in particular, if we run D on a given (z,y,v) $O(K\log K)$ times, with probability $1-\frac{1}{2K}$ we can obtain an estimate $\hat{w}(y,v)$ which differs from $w(y,v)$ by at most $\frac{1}{2K}\leq\frac{\epsilon}{4}$.

On input z, our inversion algorithm runs the $\frac{\epsilon}{4}$ soft decoding algorithm for $\mathcal{C}$ with estimated weight function $\hat{w}=\hat{w}_z$ to output a list of candidates. For each candidate c it tests if $f(c)=z$ and outputs the first such c if it exists.

We now analyze the running time. The decoding algorithm runs in time K given access to $\hat{w}$. Since each access to $\hat{w}$ is computed by $O(K\log K)$ accesses to W, each of which in turn takes time K, the total running time of the decoding algorithm is $K\cdot O(K\log K)\cdot K=O(K^3\log K)$. The candidate list has size at most K, since the decoding algorithm itself runs in time K except for accesses to $\hat{w}$. Therefore, the running time for the candidate checks is at most $K\cdot\sqrt{K^4}=K^3$. Thus the total running time is $o(K^4)$.

We now show that the algorithm inverts many inputs z. We know that D ϵ–distinguishes the distributions $(f(C),U_{[T]},h(C,U_{[T]}))$ and $(f(C),U_{[T]},U_{[M]})$. Say, without loss of generality, that D accepts $(f(C),U_{[T]},h(C,U_{[T]}))$ more often than $(f(C),U_{[T]},U_{[M]})$. I.e.,

$$\mathbf{E}_{\substack{c,y,\\ z=f(c)}} w_z(y,h(c,y)) \quad\geq\quad \mathbf{E}_{y,v}w_z(y,v)+\epsilon=\rho(w_z)+\epsilon.$$

Equivalently,

[2] Recall that ϵ–distinguishing was defined in Definition VI.1

$$\mathbf{E}_c[\frac{\mathsf{Ag}(\mathcal{C}(c), w_z)}{\mathrm{T}}] \quad \geq \quad \rho(w_z) + \epsilon.$$

An averaging argument then gives that there is a set $\mathcal{A}$ of size at least $\frac{\epsilon}{2}C$, such that for all $c \in \mathcal{A}$,

$$\frac{\mathsf{Ag}(\mathcal{C}(c), w_z)}{\mathrm{T}} \quad \geq \quad \rho(w_z) + \frac{\epsilon}{2} \qquad (4)$$

Now, say z is of the form $z = f(c)$ for $c \in \mathcal{A}$. Then, with probability at least $1 - K \cdot \frac{1}{2K} = \frac{1}{2}$, all the estimated weights $\hat{w_z}(y, v)$ will be within $\epsilon/4$ of the true weight $w_z(y, v)$. Since we output all c such that

$$\frac{1}{\mathrm{T}} \mathsf{Ag}(\mathcal{C}(c), \hat{w_z}) \geq \rho(\hat{w_z}) + \frac{\epsilon}{4},$$

if our estimates are accurate and our input is $z = f(c)$ for $c \in \mathcal{A}$, then c will appear in the candidate list. Hence, with probability at least $\frac{1}{2}$ some element of $f^{-1}(z)$ will be output. Since $\mathcal{A}$ constitutes a $\epsilon/2$ fraction of all inputs and our algorithm runs in time $o(K^4)$, this contradicts the security of f. We conclude that h is hardcore. ■

We can now use the extractor code $\mathcal{C}_{\mathrm{TR}}$ to obtain a hardcore function outputting many bits with few auxiliary random bits.

Theorem 5: There exists a constant $\alpha > 0$ such that $\mathrm{TR}_{\mathrm{c,m=\alpha k,\epsilon=\frac{1}{cM}}} : [\mathrm{C}] \times [\mathrm{T}] \to [\mathrm{M} = \mathrm{K}^{\alpha}]$, defined above in Section VII, is hardcore with security $(K = 2^k, \epsilon)$. Furthermore, the function uses only t auxiliary randomness where $t \leq O((\log c + \log(\frac{1}{\epsilon}))^2)$.

Proof: We have $\mathrm{T} \leq \mathrm{M}$ and $\epsilon = \frac{1}{\mathrm{M}}$ and so the decoding running time is bounded by $p(\mathrm{M})$ for some polynomial $p(\cdot)$. We choose M so that $p(\mathrm{M}) \leq K$, which holds as long as $m \leq \alpha k$ for some constant $\alpha < 1$, and then the decoding running time is at most K. By Theorem 4 the function TR is hardcore with security (K, ϵ). The number of truly random bits used for these parameters is $t = O(\log^2(\frac{c}{\epsilon}))$. ■

The function TR is hardcore for all one-way functions with security $(K^4, \frac{\epsilon}{4})$. We can think of a one-way function with security $(K^4, \frac{\epsilon}{4})$ as having "hardness" $4k$. We therefore see that for every such f the function TR outputs a constat fraction of the hardness of f, and does so while using only very few auxiliary random bits. Thus, the hardcore function TR uses few random bits and outputs an almost optimal number of hard bits.

Acknowledgements

We would like to thank Madhu Sudan for his suggestion to generalize list decoding results we had to soft decoding results. We would also like to thank Muli Safra, Venkatesh Srinivasan, Madhu Sudan and Umesh Vazirani for helpful discussions and comments. We would like to thank the anonymous referees for many very helpful comments to improve the presentation of the paper.

References

[1] V. Guruswami, "Better extractors for better codes?," in *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, 2004, pp. 436–444.

[2] P. Elias, "List decoding for noisy channels," in *1957-IRE WESCON Convention Record, Pt. 2*, 1957, pp. 94–104.

[3] J.M. Wozencraft, "List decoding," in *Quarterly Progress Report.* Research Laboratory of Electronics, MIT, 1958, vol. 48, pp. 90–95.

[4] O. Goldreich and L. A. Levin, "A hard-core predicate for all one-way functions," in *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, 1989, pp. 25–32.

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

[6] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometric codes," *IEEE Transactions on Information Theory*, vol. 45, pp. 1757–1767, 1999.

[7] D. Boneh, "Finding smooth integers in short intervals using CRT decoding," in *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, 2000, pp. 265–272.

[8] M. Sudan, "Coding theory: Tutorial and survey," in *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, 2001, pp. 36–53.

[9] V. Guruswami and P. Indyk, "Near-optimal linear time codes for unique decoding and new list-decodable codes over smaller alphabets," in *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, 2002, pp. 812–821.

[10] N. Nisan and D. Zuckerman, "Randomness is linear in space," *Journal of Computer and System Sciences*, vol. 52, no. 1, pp. 43–52, 1996.

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

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

[13] A. Wigderson and D. Zuckerman, "Expanders that beat the eigenvalue bound: Explicit construction and applications," *Combinatorica*, vol. 19, no. 1, pp. 125–138, 1999.

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

[15] D. Zuckerman, "On unapproximable versions of NP-complete problems," *SIAM Journal on Computing*, vol. 25, pp. 1293–1304, 1996.

[16] E. Mossel and C. Umans, "On the complexity of approximating the VC dimension," in *Proceedings of the 16th Annual IEEE Conference on Computational Complexity*, 2001, pp. 220–225.

[17] R. Shaltiel, "Recent developments in explicit constructions of extractors," *Bulletin of the European Association for Theoretical Computer Science*, vol. 77, pp. 67–95, June 2002.

[18] N. Nisan, "Extracting randomness: How and why – a survey," in *Proceedings of the 11th Annual IEEE Conference on Computational Complexity*, 1996, pp. 44–58.

[19] N. Nisan and A. Ta-Shma, "Extracting randomness: A survey and new constructions," *Journal of Computer and System Sciences*, vol. 58, pp. 148–173, 1999.

[20] S. M. Johnson, "A new upper bound for error-correcting codes," *IEEE Transactions on Information Theory*, vol. 8, pp. 203–207, 1962.

[21] L. Trevisan, "Extractors and pseudorandom generators," *Journal of the ACM*, pp. 860–879, 2001.

[22] G. D. Forney, Jr., "Generalized minimum distance decoding," *IEEE Transactions on Information Theory*, vol. 12, pp. 125–131, 1966.

[23] R. Kötter and A. Vardy, "Algebraic soft decoding of reed solomon codes," in *IEEE International Symposium on Information Theory*, 2000.

[24] V. Guruswami, A. Sahai, and M. Sudan, ""Soft-decision" decoding of Chinese remainder codes," in *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, 2000, pp. 159–168.

[25] O. Goldreich, R. Rubinfeld, and M. Sudan, "Learning polynomials with queries: The highly noisy case," *SIAM Journal on Discrete Mathematics*, vol. 13, pp. 535–570, 2000.

[26] E. Agrell, A. Vardy, and K. Zeger, "Upper bounds for constant-weight codes," *IEEE Transactions on Information Theory*, vol. 46, pp. 2373–2395, 2000.

[27] J. Radhakrishnan, ," 1997, Personal communication.

[28] R. Impagliazzo, L. A. Levin, and M. Luby, "Pseudorandom generation from one-way functions," in *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, 1989, pp. 12–24.

[29] J. Radhakrishnan and A. Ta-Shma, "Bounds for dispersers, extractors, and depth-two superconcentrators," *SIAM Journal on Discrete Mathematics*, vol. 13, no. 1, pp. 2–24, 2000.

[30] R. Shaltiel and C. Umans, "Simple extractors for all min-entropies and a new pseudo-random generator," in *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, 2001, pp. 648–657.

[31] A. Ta-Shma, "Storing information with extractors," *Information Processing Letters*, vol. 83, no. 5, pp. 267–274, 2002.

[32] H. Buhrman, P.B. Miltersen, J. Radhakrishnan, and S. Venkatesh, "Are bitvectors optimal?," in *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, 2000, pp. 449–458.

[33] R. Impagliazzo and D. Zuckerman, "How to recycle random bits," in *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, 1989, pp. 248–253.

[34] V. Guruswami, J. Hastad, M. Sudan, and D. Zuckerman, "Combinatorial bounds for list decoding," *IEEE Transactions on Information Theory*, vol. 48, pp. 1021–1034, 2002.

[35] J. Savage, *Models of Computation*, Addison-Wesley Publishing Co., 1998.

[36] N. Nisan and A. Wigderson, "Hardness vs. randomness," *Journal of Computer and System Sciences*, vol. 49, pp. 149–167, 1994.

[37] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson, "Bpp has subexponential simulation unless Exptime has publishable proofs," *Computational Complexity*, vol. 3, pp. 307–318, 1993.

[38] R. Impagliazzo and A. Wigderson, "P = BPP if E requires exponential circuits: Derandomizing the XOR lemma," in *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, 1997, pp. 220–229.

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

[40] V. Guruswami and M. Sudan, "List decoding algorithms for certain concatenated codes," in *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, 2000, pp. 181–190.

[41] R. Impagliazzo, ," July 1997, Personal communication. For a written version, see [42].

[42] M. Sudan, "List decoding: Algorithms and applications," *SIGACT News*, vol. 31, pp. 16–27, March 2000.

Biographies

Amnon Ta-Shma is a Professor in the Department of Computer Science at Tel-Aviv University. He received his B.Sc. in Mathematics and Computer Science from the Hebrew University in 1987 and his Ph.D. from the Hebrew University in 1996. He was a postdoctoral fellow at the International Computer Science Institute and the University of California at Berkeley from 1997-2000. He has been with Tel-Aviv University since then.

His research interests include the role of randomness in computation, pseudorandomness, complexity theory, coding theory, and quantum computation. His awards include Wolf Prize for Ph.D. students, Mifaal Hapais Research Prize, Alon Fellowship, and the Dan David Scholarship for Young Researchers.

David Zuckerman (M'00) is a Professor in the Department of Computer Science at the University of Texas at Austin. He received his A.B. in Mathematics from Harvard University in 1987 and his Ph.D. from the University of California at Berkeley in 1991. He was a postdoctoral fellow at MIT from 1991-1993, and at Hebrew University in the Fall of 1993. He has been with the University of Texas since then, visiting U.C. Berkeley from 1999-2000 and Harvard University from 2004-2005.

His research interests include the role of randomness in computation, pseudorandomness, complexity theory, coding theory, and cryptography. His awards include a Guggenheim Fellowship, a Radcliffe Institute Fellowship, a Packard Fellowship for Science and Engineering, a Sloan Research Fellowship, and an NSF Young Investigator Award.