The Unified Theory of Pseudorandomness*

Salil Vadhan†

October 16, 2007

Abstract

We survey the close connections between a variety of "pseudorandom objects," namely pseudorandom generators, expander graphs, list-decodable error-correcting codes, randomness extractors, averaging samplers, and hardness amplifiers.

1 Introduction

Pseudorandomness is the theory of efficiently generating objects that "look random" despite being constructed with little or no randomness. Over the past 25 years, it has developed into a substantial area of study, with significant implications for complexity theory, cryptography, algorithm design, combinatorics, and communications theory. One of the achievements of this line of work has been the realization that a number of fundamental and widely studied "pseudorandom" objects are all *almost equivalent* when viewed appropriately. These objects include:

Pseudorandom Generators These are procedures that stretch a short "seed" of truly random bits into a long string of "pseudorandom" bits that cannot be distinguished from truly random by any efficient algorithm. In this article, we focus on methods for constructing pseudorandom generators from boolean functions of high circuit complexity.

Expander Graphs Expanders are graphs that are sparse but nevertheless highly connected. There are many variants of expander graphs, but here we focus on the classical notion of *vertex expansion,* where every subset of not-too-many vertices has many neighbors in the graph.

Error-Correcting Codes These are methods for encoding messages so that even if many of the symbols are corrupted, the original message can still be recovered. Here we focus on *list decoding*, where there are so many corruptions that uniquely decoding the original message is impossible, but it is still may be possible to produce a short list of possible candidates.

Randomness Extractors These are procedures that extract almost uniformly distributed bits from sources of biased and correlated bits. Here we focus on extractors for general sources, where all we assume is a lower bound on the amount of "entropy" in the source and only get

*Also appeared in SIGACT...

†School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138. salil@eecs.harvard.edu. Supported by NSF grant CCF-0133096, ONR grant N00014-04-1-0478, and US-Israel BSF grant 2002246..

a single sample from the source. Extractors for such sources necessarily use a small number of additional truly random bits as a "seed" for extraction.

Samplers These are randomness-efficient methods for sampling elements of a large universe so that approximately the correct fraction of samples will land in any subset of the universe with high probability.

Hardness Amplifiers These are methods for converting worst-case hard boolean functions into ones that are average-case hard.

These objects are all "pseudorandom" in the sense that a randomly chosen object can be shown to have the desired properties with high probability, and the main goal is typically to find *explicit constructions* — ones that are deterministic and computationally efficient — achieving similar parameters. Each of these objects was introduced with a different motivation, and originally developed its own body of research. However, as mentioned above, research in the theory of pseudorandomness has uncovered intimate connections between all of them. In recent years, a great deal of progress has been made in understanding and constructing each of these objects by translating intuitions and techniques developed for one to the others.

The purpose of this survey is to present the connections between these objects in a single place, using a single language. Hopefully, this will make the connections more readily accessible and usable for non-experts and those familiar with some but not all of the objects at hand. In addition, it is also meant to clarify the *differences* between the objects, and explain why occasional claims of "optimal" constructions of one type of object do not always lead to improved constructions of the others.

Naturally, describing connections between six different notions in a short article makes it impossible to do justice to any of the objects in its own. Thus, for motivation, constructions, and applications, the reader is referred to existing surveys focused on the individual objects [CRT, Kab, HLW, Sud, Gur, NT, Sha, Gol1, Tre2]. For broader introductions to pseudorandomness, see [Gol2, Mil, Vad], with [Vad] emphasizing the connections described here.

The framework used in this survey extends to a number of other pseudorandom objects, such as randomness condensers, but we omit these extensions due to space constraints.

2 The Framework

Notation. For a natural number $N \in \mathbb{N}$, $[N]$ denotes the set $\{1, \ldots, N\}$. For a discrete random variable X, $x \stackrel{\mathrm{R}}{\leftarrow} X$ means that x is sampled according to X. For a set S, U_S is a random variable distributed uniformly over S. For convenience, we will sometimes write $x \stackrel{\mathrm{R}}{\leftarrow} S$ as shorthand for $x \stackrel{\mathrm{R}}{\leftarrow} U_S$. All logs are base 2.

As we will see, all of the objects we are discussing can be syntactically viewed as functions $\Gamma : [N] \times [D] \to [M]$. We will show how the defining properties of each of the objects can be cast in terms of the following notion.

Definition 1 *For a function $\Gamma : [N] \times [D] \to [M]$, a set $T \subseteq [M]$, and an agreement parameter $\varepsilon \in [0, 1)$, we define*

$$\mathrm{LIST}_\Gamma(T, \varepsilon) \quad = \quad \{x \in [N] : \Pr[\Gamma(x, U_{[D]}) \in T] > \varepsilon\}$$

We also define $\mathrm{LIST}_\Gamma(T,1) = \{x \in [N] : \Pr[\Gamma(x, U_{[D]}) \in T] = 1\}$.

In general, it will be possible to characterize each of the pseudorandom objects by a condition of the following form:

$$\text{“For every subset } T \in \mathcal{C}, \text{ we have } |\mathrm{LIST}_\Gamma(T,\varepsilon)| \leq K.\text{”}$$

Here $\varepsilon \in [0,1]$ and $K \in [0,N]$ will be parameters corresponding to the "quality" of the object, and we usually wish to minimize both. $\mathcal{C}$ will be a class of subsets of $[M]$, sometimes governed by an additional "quality" parameter. Sometimes the requirement will be that the size of $\mathrm{LIST}_\Gamma(T,\varepsilon)$ is *strictly* less than K, but this is just a matter of notation, amounting to replacing K in the above formulation by $\lceil K \rceil - 1$.

The notation "$\mathrm{LIST}_\Gamma(\cdot,\cdot)$" comes from the interpretation of list-decodable error-correcting codes in this framework (detailed in the next section), where T corresponds to a corrupted codeword and $\mathrm{LIST}_\Gamma(T,\varepsilon)$ to the list of possible decodings. This list-decoding viewpoint turns out to be very useful for casting all of the objects in the same language. However, this is not the only way of looking at the objects, and indeed the power of the connections we describe in this survey is the variety of perspectives they provide. In particular, many of the connections were discovered through the study of randomness extractors, and extractors remain a powerful lens through which to view the area. The list-decoding view of extractors, and consequently of many of the other objects presented here, emerged through a sequence of works, and was crystalized in paper of Ta-Shma and Zuckerman [TZ].

Our notation (e.g. the parameters N,M,D, K, ε) follows the literature on extractors, and thus is nonstandard for some of the objects. We also follow the convention from the extractor literature that $n = \log N$, $d = \log D$, $m = \log M$, and $k = \log K$. While it is not necessary for the definitions to make sense, in some cases it is more natural to think of N, D, and/or M as a power of 2, and thus the sets $[N]$, $[D]$, and $[M]$ as corresponding to the set of bit-strings of length n, d, and m, respectively. In some cases (namely, list-decodable codes and hardness amplifiers), we will restrict to functions in which y is a prefix of $\Gamma(x,y)$, and then it will be convenient to denote the range by $[D] \times [q]$ rather than $[M]$. This syntactic constraint actually leads to natural variants (sometimes referred to as "strong" variants) of the other objects, too, but we do not impose it here for sake of generality and consistency with the most commonly used definitions.

Much of the work on the objects are discussing is concerned with giving *explicit* constructions, which correspond to the function $\Gamma : [N] \times [D] \to [M]$ being deterministically and efficiently computable, e.g. in time $\mathrm{poly}(n,d)$. However, since our focus is on the connections between the objects rather than their constructions, we will generally not discuss explicitness except in passing.

3 List-Decodable Codes

We begin by describing how the standard notion of list-decodable codes can be cast in the framework, because it motivates the notation $\mathrm{LIST}_\Gamma(\cdot,\cdot)$ and provides a good basis for understanding the other objects.

A *code* is a function mapping n-bit messages to codewords consisting of D symbols over an alphabet of size q. More generally, it can be a function $\mathrm{ECC} : [N] \to [q]^D$. (In the coding literature, the message length is denoted by k rather than n and the codeword length is n rather than D.) The goal is to define the function ECC so that if a codeword $\mathrm{ECC}(x)$ is corrupted in a significant number of symbols and one only receives the corrupted string $r \in [q]^D$, the message x can still be recovered. *List-decodable* codes are designed for a setting where the number of corruptions is too large to hope for uniquely decoding x, and thus we settle for getting a short list of possible candidates.

Definition 2 *A code* ECC : $[N] \to [q]^D$ *is* (ε, K) list-decodable *if for every "received word"* $r \in [q]^D$*, there are at most* K *messages* $x \in [N]$ *such that* $\mathrm{ECC}(x)$ *and* r *agree in greater than a* $1/q + \varepsilon$ *fraction of positions.*

This definition says that if we receive a string $r \in [q]^D$ that we know has resulted from corrupting a codeword $\mathrm{ECC}(x)$ in less than a $1 - (1/q + \varepsilon)$ fraction of positions, then we can pin down the message x to one of at most K possibilities. K is thus called the *list size*. Note that we expect a uniformly random string $r \xleftarrow{\mathrm{R}} [q]^D$ to agree with most codewords in roughly a $1/q$ fraction of positions so we cannot expect to do any meaningful decoding from agreement $1/q$; this is why we ask for agreement greater than $1/q + \varepsilon$.

Naturally, one wants the agreement parameter ε to be as small possible and the (relative) *rate* $\rho = n/dm = \log N/(D \log q)$ of the code to be as large as possible.

In coding theory, one typically considers both ε and ρ to be fixed constants in $(0, 1)$, while the message length $n = \log N$ tends to infinity and the alphabet size remains small (ideally, $q = O(1)$). The main challenge is to achieve an optimal tradeoff between the rate and agreement, while maintaining a list size K polynomially bounded in the message length n. Indeed, we usually also want an efficient algorithm that enumerates all the possible decodings x in time polynomial in n, which implies a polynomial bound on the list size. There has been dramatic progress on this challenge in the recent years; see the surveys [Sud, Gur].

To cast list-decodable codes in our framework, note that given a code ECC : $[N] \to [q]^D$, we can define a function $\Gamma : [N] \times [D] \to [D] \times [q]$ by

$$\Gamma(x, y) = (y, \mathrm{ECC}(x)_y). \tag{1}$$

Note that the range of Γ is $[D] \times [q]$ and it has the property that the first component of $\Gamma(x, y)$ is always y. Moreover, given any Γ with this property, we can obtain a corresponding code ECC.

Proposition 3 *Let the code* ECC : $[N] \to [q]^D$ *correspond to the function* $\Gamma : [N] \times [D] \to [D] \times [q]$ *via (1). Then* ECC *is* (ε, K) *list-decodable if and only if*

$$\forall r \in [q]^D \qquad |\mathrm{LIST}_\Gamma(T_r, 1/q + \varepsilon)| \le K,$$

where $T_r = \{(y, r_y) : y \in [D]\}$.

Proof: It suffices to show that for every $r \in [q]^D$ and $x \in [N]$, we have $x \in \mathrm{LIST}_\Gamma(T_r, 1/q + \varepsilon)$ iff $\mathrm{ECC}(x)$ agrees with r in greater than a $1/q + \varepsilon$ fraction of places. We show this as follows:

$$\begin{aligned} x \in \mathrm{LIST}_\Gamma(T_r, 1/q + \varepsilon) \quad &\Leftrightarrow \quad \Pr_{y \xleftarrow{\mathrm{R}} [D]} [\Gamma(x, y) \in T_r] > 1/q + \varepsilon \\ &\Leftrightarrow \quad \Pr_{y \xleftarrow{\mathrm{R}} [D]} [(y, \mathrm{ECC}(x)_y) \in T_r] > 1/q + \varepsilon \\ &\Leftrightarrow \quad \Pr_{y \xleftarrow{\mathrm{R}} [D]} [\mathrm{ECC}(x)_y = r_y] > 1/q + \varepsilon \end{aligned}$$

■

In addition to the particular range of parameters typically studied (e.g. the small alphabet size q), the other feature that distinguishes list-decodable codes from many of the other objects described below is that it only considers sets of the form $T_r \subseteq [D] \times [q]$ for received words $r \in [q]^D$. These sets contain only one element for each possible first component $y \in [D]$, and thus are of size exactly D. Note that as the alphabet size q grows, these sets contain a vanishingly small fraction of the range $[D] \times [q]$.

4 Samplers

Suppose we are interested in estimating the average value of a boolean function T on a huge domain $[M]$, given an oracle for T. The Chernoff Bound tells us that if we take $D = O(\log(1/\delta)/\varepsilon^2)$ independent random samples from $[M]$, then with probability at least $1 - \delta$, the average of T on the sample will approximate T's global average within an additive error of ε. However, it is well known that these samples need not be generated independently; for example, samples generated according to a k-wise independent distribution or by a random walk on an expander graph have similar properties [CG2, BR, SSS, Gil]. The advantage of using such correlated sample spaces is that the samples can be generated using many fewer random bits than independent samples; this can be useful for derandomization and or simply because it provides a compact representation of the sequence of samples.

The definition below abstracts this idea of a procedure that uses $n = \log N$ random bits to generate D samples from $[M]$ with the above average-approximation property.

Definition 4 ([BR][1]) *A* sampler Smp *for domain size M is given "coin tosses" $x \xleftarrow{R} [N]$ and outputs samples $z_1, \ldots, z_D \in [M]$. We say that* Smp *is a (δ, ε)* averaging sampler *if for every function $T : [M] \to \{0, 1\}$, we have*

$$\Pr_{(z_1,\ldots,z_D) \xleftarrow{R} \mathrm{Smp}(U_{[N]})} \left[\frac{1}{D} \sum_i T(z_i) \leq \mu(T) + \varepsilon \right] \geq 1 - \delta,$$

where $\mu(T) \stackrel{\text{def}}{=} \mathrm{E}[T(U_{[M]})]$.

Note that the definition only bounds the probability that the sample-average deviates from $\mu(T)$ from above. However, a bound in both directions can be obtained by applying the above definition also to the complement of T, at the price of a factor of 2 in the error probability δ. (Considering deviations in only one direction will allow us to cast samplers in our framework without any slackness in parameters.) We note that the above definition can be also generalized to functions T that are not necessarily boolean, and instead map to the real interval $[0, 1]$. Non-boolean samplers and boolean samplers turn out to be equivalent up to a small loss in the parameters [Zuc2].

We note that one can consider more general notions of samplers that make adaptive oracle queries to the function T and/or produce their estimate of $\mu(T)$ by an arbitrary computation on the values returned (not necessarily taking the sample average). In fact, utilizing this additional flexibility, there are known explicit samplers that achieve better parameters than we know how to achieve with averaging samplers. (For these generalizations, constructions of such samplers, and discussion of other issues regarding samplers, see the survey [Gol1].) Nevertheless, some applications require averaging samplers, and averaging samplers are also more closely related to the other objects we are studying.

In terms of the parameters, one typically considers M, ε, and δ as given, and seeks to minimize both the number $n = \log N$ of random bits and the number D of samples. Usually, complexity is measured as a function of $m = \log M$, with ε ranging between constant and $1/\mathrm{poly}(m)$, and δ ranging between $o(1)$ and $2^{-\mathrm{poly}(m)}$.

Samplers can be cast rather directly into our framework as follows. Given a sampler Smp for domain size M that generates D samples using coin tosses from $[N]$, we can define $\Gamma : [N] \times [D] \to [M]$ by setting

$$\Gamma(x, y) = \text{the } y\text{'th sample of Smp on coin tosses } x. \tag{2}$$

Conversely, any function $\Gamma : [N] \times [D] \to [M]$ yields a sampler. The property of Smp being an averaging sampler can be translated to the "list-decodability" of Γ as follows.

Proposition 5 *Let* Smp *be a sampler for domain size M that generates D samples using coin tosses from $[N]$, and let $\Gamma : [N] \times [D] \to [M]$ be the function corresponding to* Smp *via (2). Then* Smp *is a (δ, ε) averaging sampler if and only if*

$$\forall T \subseteq [M] \qquad |\mathrm{LIST}_\Gamma(T, \mu(T) + \varepsilon)| \leq K,$$

where $K = \delta N$ and $\mu(T) \stackrel{\mathrm{def}}{=} |T|/M$.

Proof: We can view a function $T : [M] \to \{0, 1\}$ as the characteristic function of a subset of $[M]$, which, by abuse of notation, we also denote by T. Note that $\mathrm{LIST}_\Gamma(T, \mu(T) + \varepsilon)$ is precisely the set of coin tosses x for which $\mathrm{Smp}(x)$ outputs a sample on which T's average is greater than $\mu(T) + \varepsilon$. Thus, the probability of a bad sample is at most δ iff $|\mathrm{LIST}_\Gamma(T, \mu(T) + \varepsilon)| \leq \delta N$. ■

Let's compare the characterization of samplers given by Proposition 5 to the characterization of list-decodable codes given by Proposition 3. One difference is that codes correspond to functions Γ where $\Gamma(x, y)$ always includes y as a prefix. This turns out to be a relatively minor difference, and most known samplers can be modified to have this property. A major difference, however, is that for list-decodable codes, we only consider decoding from sets of the form T_r for some received word $r \in [q]^D$. Otherwise, the two characterizations are identical. (Note that $\mu(T_r) = 1/q$, and bounding K and bounding δ are equivalent via the relation $K = \delta N$.) Still, the settings of parameters typically considered in the two cases are quite different. In codes, the main growing parameter is the message length $n = \log N$, and one typically wants the alphabet size q to be a constant (e.g. $q = 2$) and the codeword length D to be linear in n. Thus, the range of Γ is of size $M = D \cdot q = O(\log N)$. In samplers, the main growing parameter is $m = \log M$, which is the number of random bits needed to select a single sample of the universe $[M]$ uniformly at random, and one typically seeks samplers using a number random bits $n = \log N$ that is linear (or polynomial) in m. Thus, $M = N^{\Omega(1)}$, in sharp contrast to the typical setting for codes). Also in contrast to codes, samplers are interesting even when δ is a constant independent of N (or vanishes slowly as a function of N). In such a case, the number of samples can can be independent of N (e.g. in an optimal sampler, $D = O(\log(1/\delta)/\varepsilon^2)$. But constant δ in codes means that the list size $K = \delta N$ is a constant fraction of the message space, which seems too large to be useful from a coding perspective. Instead, the list size for codes is typically required to be $K = \mathrm{poly}(n) = \mathrm{poly}(\log N)$, which forces the the codeword length D to be as large as the message length $n = \log N$.

5 Expander Graphs

Expanders are graphs with two seemingly contradictory properties. On one hand, they have very low degree; on the other, they are extremely well-connected. Expanders have numerous applications in theoretical computer science, and their study has also turned out to be mathematically very rich; see the survey [HLW].

There are a variety of measures of expansion, with close relationships between them, but here we will focus on the most basic measure, known as *vertex expansion*. We restrict attention to bipartite graphs, where the requirement is that every set of left-vertices that is not too large must have "many" neighbors on the right. We allow multiple edges between vertices. We require the graph to be left-regular, but it need not be right-regular.

Definition 6 *Let G be a left-regular bipartite multigraph with left vertex set $[N]$, right vertex set $[M]$, and left-degree D. G is an $(=K, A)$* expander *if every left-set S of size at least K has at least $A \cdot K$ neighbors on the right. G is a (K, A)* expander *if it is a $(=K', A)$* expander *for every $K' \le K$.*

The classic setting of parameters for expanders is the balanced one, where $M = N$, and then the goal is to have the degree D and the expansion factor A to both be constants independent of the number of vertices, with $A > 1$ and expansion achieved for sets of size up to $K = \Omega(M)$. However, the imbalanced case $M < N$ is also interesting, and then even expansion factors A smaller than 1 are nontrivial (provided $A > M/N$).

We can cast expanders in our framework as follows. For a left-regular bipartite multigraph G with left vertex set $[N]$, right vertex set $[M]$, and left-degree D, we define the *neighbor function* $\Gamma : [N] \times [D] \to [M]$ by

$$\Gamma(x, y) = \text{the } y\text{'th neighbor of } x \tag{3}$$

Proposition 7 *Let G be a left-regular bipartite multigraph with left vertex set $[N]$, right vertex set $[M]$, and left-degree D, and let $\Gamma : [N] \times [D] \to [M]$ be the neighbor function corresponding to G via (3). Then G is an $(=K, A)$ expander if and only if*

$$\forall T \subseteq [M] \text{ s.t. } |T| < AK \qquad |\mathrm{LIST}_\Gamma(T, 1)| < K. \tag{4}$$

Thus, G is a (K, A) expander iff for every $T \subseteq [M]$ of size less than AK, we have $|\mathrm{LIST}_\Gamma(T, 1)| < |T|/A$.

Proof: We show that G fails to be an $(=K, A)$ expander iff Condition (4) is false.

If G is not an $(=K, A)$ expander, then there is a left-set $S \subseteq [N]$ of size at least K with fewer than AK neighbors on the right. Let T be the set of neighbors of S. Then $|T| < AK$ but $S \subseteq \mathrm{LIST}_\Gamma(T, 1)$, so $|\mathrm{LIST}_\Gamma(T, 1)| \ge K$, violating Condition (4).

Conversely, suppose that Condition (4) fails. Then there is a right-set $T \subseteq [M]$ of size less than AK for which $|\mathrm{LIST}_\Gamma(T, 1)| \ge K$. But the neighbors of $\mathrm{LIST}_\Gamma(T, 1)$ are all elements of T, violating expansion. ■

We now compare the characterization of expanders given in Proposition 7 to those for list-decodable codes and samplers. First, note that we quantify over all sets T of a bounded size (namely, smaller than AK). In codes, the sets T were also of a small size but also restricted to be of the form T_r for a received word r. In samplers, there was no constraint on T. Second, we only need a bound on $|\mathrm{LIST}_\Gamma(T,1)|$, which is conceivably easier to obtain than a bound on $|\mathrm{LIST}_\Gamma(T,\mu(T)+\varepsilon)|$ as in codes and samplers. Nevertheless, depending on the parameters, vertex expansion (as in Definition 6 and Proposition 7) often implies stronger measures of expansion (such as spectral gap [Alo] and randomness condensing [TUZ]), which in turn imply bounds on $|\mathrm{LIST}_\Gamma(T,\mu(T)+\varepsilon)|$.

The typical parameter ranges for expanders are more similar to those for samplers than for those of codes. Specifically, N and M tend to be of comparable size; indeed, the classic case is $N = M$, and even in the unbalanced case, they are typically polynomially related. However, for expanders, there is no parameter ε. On the other hand, there is something new to optimize, namely the expansion factor A, which is the ratio between the size of T and the list size K. In particular, to have expansion factor larger than 1 (the classic setting of parameters for expansion), we must have a list size that is *smaller* than $|T|$. In list-decodable codes, the distinction between list size $D = |T_r|$ and, say, $D/2$ seems to be less significant.

Despite the above differences, recent constructions of list-decodable codes have proved useful in constructing expanders with near-optimal expansion factors (namely, $A = (1-\varepsilon)D$) via Proposition 7 [GUV]. A formulation of expansion similar to Proposition 7 also appeared in [GT].

6 Randomness Extractors

A randomness extractor is a function that extracts almost-uniform bits from a source of biased and correlated bits. Their original motivation was the simulation of randomized algorithms with physical sources of randomness, but they have turned out to have a wide variety of other applications in theoretical computer science. Moreover, they have played a unifying role in the theory of pseudorandomness, and have been the avenue through which many of the connections described in this survey were discovered. The history, applications, and constructions of extractors are described in more detail in [NT, Sha].

To formalize the notion of an extractor, we need to model a "source of biased and correlated bits" and define what it means for the output of the extractor to be "almost uniform." For the former, we adopt a very general notion, advocated in [CG1, Zuc1], where we only require that the source has enough randomness in it, as measured by the following variant of entropy.

Definition 8 *The* min-entropy *of a random variable X is $\mathrm{H}_\infty(X) = \min_{x\in\mathrm{Supp}(X)} \log(1/\Pr[X = x])$. X is a k*-source *if $\mathrm{H}_\infty(X) \geq k$. Equivalently, X is a k*-source *if $\Pr[X = x] \leq 2^{-k}$ for all x.*

Intuitively, we think of a k-source as having "k bits of randomness" in it. For example, a random variable that is uniformly distributed over any $K = 2^k$ strings is a k-source.

For the quality of the output of the extractor, we use the standard measure closeness between probability distributions.

Definition 9 *The* statistical difference *between random variables X and Y taking values in a universe $[M]$ is defined to be*

$$\Delta(X,Y) = \max_{T\subseteq[M]} |\Pr[X\in T] - \Pr[Y \in T]| = \max_{T\subseteq[M]} \Pr[X\in T] - \Pr[Y\in T].$$

X and Y are ε-close *if $\Delta(X,Y) \leq \varepsilon$. Otherwise, we say they are* ε-far.

The equivalence between the formulations of statistical difference with and without the absolute values can be seen by observing that $\Pr[X \in T] - \Pr[Y \in T] = -(\Pr[X \in \overline{T}] - \Pr[Y \in \overline{T}])$.

Ideally we'd like an extractor to be a function $\mathrm{Ext} : [N] \to [M]$ such that for every k-source X taking values in $[N]$, the random variable $\mathrm{Ext}(X)$ is ε-close to $U_{[M]}$. That is, given an n-bit string coming from an unknown random source with at least k bits of randomness, the extractor is guaranteed to produce m bits that are close to uniform. However, this is easily seen to be impossible even when $m = 1$: the uniform distribution on either $\mathrm{Ext}^{-1}(0)$ or $\mathrm{Ext}^{-1}(1)$ is an $(n-1)$-source on which the output of the extractor is constant.

Nisan and Zuckerman [NZ] proposed to get around this difficulty by allowing the extractor a small number of truly random bits as a *seed* for the extraction.[2] This leads to the following definition.

Definition 10 ([NZ]) $\mathrm{Ext} : [N] \times [D] \to [M]$ *is a* (k, ε) extractor *if for every k-source X taking values in* $[N]$, $\mathrm{Ext}(X, U_{[D]})$ *is ε-close to* $U_{[M]}$.

The reason extraction is still interesting is that the number $d = \log D$ of truly random bits can be much smaller than the number of almost-uniform bits extracted. Indeed, d can be even be logarithmic, and thus in many applications, the need for a seed can be eliminated by enumerating all 2^d possibilities.

The ranges of min-entropy threshold k most commonly studied in the extractor literature are $k = \alpha n$ or $k = n^\alpha$ for constants $\alpha \in (0, 1)$, where $n = \log N$ is the length of the source. The error parameter ε is often taken to be a small constant, but vanishing ε is important for some applications (especially in cryptography). One usually aims to have a seed length $d = O(\log n)$ or $d = \mathrm{polylog}(n)$, and have the output length $m = \log M$ be as close to k as possible, corresponding to extracting almost all of the randomness from the source. (Ideally, $m \approx k + d$, but $m = \Omega(k)$ or $m = k^{\Omega(1)}$ often suffices.)

Notice that the syntax of extractors already matches that of the functions $\Gamma : [N] \times [D] \to [M]$ studied in our framework. The extraction property can be captured, with a small slackness in parameters, as follows.

Proposition 11 *Let* $\Gamma = \mathrm{Ext} : [N] \times [D] \to [M]$ *and let* $K = 2^k$. *Then:*

1. *If* Ext *is a* (k, ε) *extractor, then*

$$\forall T \subseteq [M] \qquad |\mathrm{LIST}_\Gamma(T, \mu(T) + \varepsilon)| < K, \tag{5}$$

   where $\mu(T) = |T|/M$.

2. *Conversely, if Condition 5 holds, then* Ext *is a* $(k + \log(1/\varepsilon), 2\varepsilon)$ *extractor.*

Proof:

1. Suppose that Condition (5) fails. That is, there is a set $T \subseteq [M]$ such that $|\mathrm{LIST}_\Gamma(T, \mu(T) + \varepsilon)| \geq K$. Let X be a random variable distributed uniformly over $\mathrm{LIST}_\Gamma(T, \mu(T) + \varepsilon)$. Then

[2] Another way around the difficulty is to consider more restricted classes of sources or to allow multiple *independent* sources. There is a large and beautiful literature on "deterministic" extractors for these cases, which we do not discuss here.

X is a k-source, but

$$\begin{aligned}\Pr[\mathrm{Ext}(X,U_{[D]})\in T] &= \mathop{\mathrm{E}}_{x\xleftarrow{\mathrm{R}}X}\left[\Pr[\mathrm{Ext}(x,U_{[D]})\in T]\right]\\ &> \mu(T)+\varepsilon\\ &= \Pr[U_{[M]}\in T]+\varepsilon,\end{aligned}$$

so $\mathrm{Ext}(X,U_{[D]})$ is ε-far from $U_{[M]}$. Thus, Ext is not a (k,ε) extractor.

2. Suppose Condition (5) holds. To show that Ext is a $(k+\log(1/\varepsilon),2\varepsilon)$ extractor, let X be any $(k+\log(1/\varepsilon))$-source taking values in $[N]$. We need to show that $\mathrm{Ext}(X,U_{[D]})$ is 2ε-close to $U_{[M]}$. That is, we need to show that for every $T\subseteq[M]$, $\Pr[\mathrm{Ext}(X,U_{[D]})\in T]\le\mu(T)+2\varepsilon$.

   So let T be any subset of $[M]$. Then

$$\begin{aligned}&\Pr[\mathrm{Ext}(X,U_{[D]})\in T]\\ &\quad\le \Pr[X\in\mathrm{LIST}(T,\mu(T)+\varepsilon)]+\Pr[\mathrm{Ext}(X,U_{[D]})\in T|X\notin\mathrm{LIST}(T,\mu(T)+\varepsilon)]\\ &\quad\le |\mathrm{LIST}(T,\mu(T)+\varepsilon)|\cdot 2^{-(k+\log(1/\varepsilon))}+(\mu(T)+\varepsilon)\\ &\quad\le K\cdot 2^{-(k+\log(1/\varepsilon))}+\mu(T)+\varepsilon\\ &\quad= \mu(T)+2\varepsilon\end{aligned}$$

■

The slackness in parameters in the above characterization is typically insignificant for extractors. Indeed, it is known that extractors must lose at least $\Theta(\log(1/\varepsilon))$ bits of the source entropy [RT], and the above slackness only affects the leading constant.

Notice that the condition characterizing extractors here is *identical* to the one characterizing averaging samplers in Proposition 5. Thus, the only real difference between extractors and averaging samplers is one of perspective, and both perspectives can be useful. For example, recall that in samplers, we measure the error probability $\delta=K/N=2^k/2^n$, whereas in extractors we measure the min-entropy threshold k on its own. Thus, the sampler perspective can be more natural when δ is relatively large compared to $1/N$, and the extractor perspective when δ becomes quite close to $1/N$. Indeed, an extractor for min-entropy $k=o(n)$ corresponds to a sampler with error probability $\delta=1/2^{(1-o(1))n}$, which means that each of the n bits of randomness used by the sampler reduces the error probability by almost a factor of 2!

This connection between extractors and samplers was proven and exploited by Zuckerman [Zuc2]. The characterization of extractors in Proposition 11 was implicit in [Zuc2, Tre1], and was explicitly formalized in coding-theoretic terms by Ta-Shma and Zuckerman [TZ].

7 Hardness Amplifiers

The connections described in following two sections, which emerged from the work of Trevisan [Tre1], are perhaps the most surprising of all, because they establish a link between complexity-theoretic objects (which refer to intractability and circuit complexity) and the purely information-theoretic and combinatorial objects we have been discussing so far.

Hardness amplification is the task of increasing the average-case hardness of a function. We measure the average-case hardness of a function by the fraction of inputs on which every efficient algorithm (or circuit) must err.

Definition 12 *A function* $f : [n] \to [q]$ *is* (δ, s)-hard *if for every boolean circuit* C *of size* s*, we have*

$$\Pr[C(U_{[n]}) \neq f(U_{[n]})] > \delta.$$

Hardness amplification is concerned with transforming a function so as to increase δ, the fraction of inputs on which it is hard. Ideally, we would like to go from $\delta = 0$, corresponding to *worst-case* hardness, to $\delta = 1 - 1/q - \varepsilon$, which is the largest value we can hope for (since every function with a range of $[q]$ can be computed correctly on a $1/q$ fraction of inputs by a constant circuit). In addition to the basic motivation of relating worst-case and average-case hardness, such hardness amplifications also are useful in constructing pseudorandom generators (see Section 8), because it is easier to construct pseudorandom generators from average-case hard functions (specifically, when $q = 2$ and $\delta = 1/2 - \varepsilon$) [NW, BFNW].

To make the goal more precise, we are interested in transformations for converting a function $f : [n] \to \{0, 1\}$ that is $(0, s)$ hard to a function $f' : [n'] \to [q]$ that is $(1 - 1/q - \varepsilon, s')$ hard for a constant q (ideally $q = 2$) and small ε. (The restriction of f to have range $\{0, 1\}$ is without loss of generality when considering worst-case hardness; otherwise we can use the function that outputs the j'th bit of $f(i)$ on input (i, j).)

The price that we usually pay for such hardness amplifications is that the circuit size for which the function is hard decreases (i.e. $s' < s$) and the domain size increases (i.e. $n' > n$); we would like these to be losses to be moderate (e.g. polynomial). Also, the complexity of computing the function correctly often increases and we again would like this increase to be moderate (e.g. f' should be computable in exponential time if f is). However, this latter property turns out to correspond to the "explicitness" of the construction, and thus we will not discuss it further below.

Several transformations achieving the above goal of converting worst-case hardness into average-case hardness are known; see the surveys [Kab, Tre2]. Like most (but not all!) results in complexity theory, these transformations are typically "black box" in the following sense. First, a single "universal" transformation algorithm Amp is given that shows how to compute f' given oracle access to f, and this transformation is well-defined for every oracle f, regardless of its complexity (even though we are ultimately interested only in functions f' within some complexity class, such as exponential time). Second, the property that f' is average-case hard when f is worst-case hard is proven by giving an "reduction" algorithm Red that efficiently converts algorithms r computing f' well on average into algorithms computing f in the worst-case. (Thus if f is hard in the worst case, there can be no efficient r computing f' well on average.) Again, even though we are ultimately interested in applying the reduction to efficient algorithms r, this property of the reduction should hold given any oracle r, regardless of its efficiency. Since our notion of hardness refers to nonuniform circuits, we will allow the reduction Red to use some nonuniform advice, which may depend on both f and r.

Black-box worst-case-to-average-case hardness amplifiers as described here are captured by the following definition.

Definition 13 *Let* $\mathrm{Amp}^f : [D] \to [q]$ *be an algorithm that is defined for every oracle* $f : [n] \to \{0, 1\}$*. We say that* Amp *is a* (t, k, ε) black-box worst-case-to-average-case hardness amplifier *if there is an oracle algorithm* Red*, called the* reduction*, running in time* t *such that for every function* $r : [D] \to [q]$ *such that*

$$\Pr[r(U_{[D]}) = \mathrm{Amp}^f(U_{[D]})] > 1/q + \varepsilon,$$

there is an advice string $z \in [K]$, where $K = 2^k$, such that

$$\forall i \in [n] \qquad \mathrm{Red}^r(i, z) = f(i).$$

The amplified function is $f' = \mathrm{Amp}^f$; we have denoted the domain size as D rather than n' for convenience below. Note that without loss of generality, $k \leq t$, because an algorithm running in time t cannot read more than t bits of its advice string.

The following proposition shows that transformations meeting Definition 13 suffice for amplifying hardness.

Proposition 14 *If* Amp *is a (t, k, ε) black-box hardness amplifier and f is $(0, s)$ hard, then* Amp^f *is $(1 - 1/q - \varepsilon, s/\tilde{O}(t))$ hard.*

Proof: Suppose for contradiction there is a circuit $r : [D] \to [q]$ of size s' computing Amp^f on greater than a $1 - 1/q + \varepsilon$ fraction of inputs. Then there is an advice string z such that $\mathrm{Red}^r(\cdot, z)$ computes f correctly on all inputs. Hardwiring z and using the fact that algorithms running in time t can be simulated by circuits of size $\tilde{O}(t)$, we get a circuit of size $\tilde{O}(t) \cdot s'$ computing f correctly on all inputs. This is a contradiction for $s' = s/\tilde{O}(t)$. ■

Typical settings of parameters for hardness amplification are $q = 2$, ε ranging from $o(1)$ to $1/n^{\Omega(1)}$, and $t = \mathrm{poly}(\log n, 1/\varepsilon)$. Note that we make no reference to the length k of the advice string, and it does not appear in the conclusion of Proposition 14. Indeed, for the purposes of hardness amplification against nonuniform circuits, k may as well be set equal to running time t of the reduction. However, below it will be clarifying to separate these two parameters.

Now we place black-box hardness amplifiers in our framework. Given $\mathrm{Amp}^f : [D] \to [q]$ defined for every oracle $f : [n] \to \{0, 1\}$, we can define $\Gamma : [N] \times [D] \to [D] \times [q]$ by

$$\Gamma(f, y) = (y, \mathrm{Amp}^f(y)), \tag{6}$$

where $N = 2^n$ and we view $[N]$ as consisting of all boolean functions on $[n]$. Just as with list-decodable codes, the second input y is a prefix of the output of Γ. Moreover, any function Γ with this property yields a corresponding amplification algorithm Amp in the natural way. This syntactic similarity between codes and hardness amplifiers is no coincidence. The next proposition shows that, if we allow reductions Red of *unbounded* running time t (but still bounded advice length k), then black-box hardness amplifiers are *equivalent* to list-decodable codes.

Proposition 15 *Let* $\mathrm{Amp}^f : [D] \to [q]$ *be an algorithm that is defined for every oracle* $f : [n] \to \{0, 1\}$. *Let* $\Gamma : [N] \times [D] \to [D] \times [q]$ *be the function corresponding to* Amp *via (6), where* $N = 2^n$. *Then* Amp *is an* (∞, k, ε) *black-box hardness amplifier if and only if*

$$\forall r \in [q]^D \qquad |\mathrm{LIST}_\Gamma(T_r, 1/q + \varepsilon)| \leq K,$$

where $T_r = \{(y, r_y) : y \in [D]\}$.

Note that the characterization given here is indeed identical to that of list-decodable codes given in Proposition 3.

Proof: First note that, viewing strings $r \in [q]^D$ as functions $r : [D] \to [q]$, we have $f \in \mathrm{LIST}_\Gamma(T_r, 1/q + \varepsilon)$ iff

$$\Pr[r(U_{[D]}) = \mathrm{Amp}^f(U_{[D]})] > 1/q + \varepsilon. \qquad (7)$$

So we need to show that Amp is an (∞, k, ε) black-box hardness amplifier if and only if, for every function $r : [D] \to [q]$, there are at most K functions f satisfying Inequality (7).

Suppose that Amp is an (∞, k, ε) black-box hardness amplifier, let Red be the associated reduction, and let $r : [D] \to [q]$ be any function. If a function f satisfies Inequality (7), then, by Definition 13, f is of the form $\mathrm{Red}^r(\cdot, z)$ for some $z \in [K]$. Since there are at most K choices for z, there are at most K functions satisfying Inequality (7).

Conversely, suppose that for every function $r : [D] \to [q]$, there are at most K functions f satisfying Inequality (7). Let $f_{r,1}, \ldots, f_{r,K}$ be these functions in lexicographic order (repeating the last one if necessary to have exactly K functions). Then we can define the reduction Red by $\mathrm{Red}^r(i, z) = f_{r,z}(i)$. (Recall that Red has unbounded running time, so constructing the list $f_{r,1}, \ldots, f_{r,K}$ can be done by brute force.) By construction, for every function f satisfying Inequality (7), there exists a $z \in [K]$ such that $\mathrm{Red}^r(\cdot, z) = f(\cdot)$. Thus Amp is an (∞, k, ε) black-box amplifier. ■

What about black-box amplifiers with reductions of *bounded* running time, as are needed for complexity-theoretic applications? First, note that every (t, k, ε) amplifier is also an (∞, k, ε) amplifiers, so we conclude that black-box amplifiers with efficient reductions are stronger than list-decodable codes. However, the efficiency of the reduction does have a natural coding-theoretic interpretation. Combining Constructions 1 and 6, we can interpret the role of the reduction Red in the following manner.

Assume for starters that Red does not use any advice, i.e. $K = 1$. Then Red is given oracle access to a received word $r \in [q]^D$ (meaning that it can ask for the j'th symbol of r in one time step) and is given a message coordinate $i \in [n]$, and should output the i'th symbol of the message f in time t. This is precisely the notion of *local decoding* for an error-correcting code; see the survey [Tre2]. Normally, a decoding algorithm is given the received word r in its entirety and should output the corresponding message f (or the list of possible messages f) in its entirety, ideally in polynomial time (e.g. time $\mathrm{poly}(D, \log q) \geq \mathrm{poly}(n)$). Here, however, we are interested in much smaller running times, such as $t = \mathrm{poly}(\log n, 1/\varepsilon)$, so the decoder does not even have time to read the entire received word or write the entire message. Instead we give it oracle access to the received word and only ask to decode a particular message symbol in which we are interested.

From the previous paragraph, we see that black-box worst-case-to-average-case hardness amplifiers with no advice and bounded running time are *equivalent* to locally decodable error-correcting codes. With advice, Definition 13 provides a natural formulation of locally *list*-decodable error-correcting codes, where the number k of advice bits corresponds to the list size $K = 2^k$. (It is sometimes useful to allow a more general formulation, where the correspondence between the advice strings z and decodings f, can be determined by a randomized preprocessing phase, which is oracle access to r; see [STV].)

Despite their close relationship, there are some differences in the typical parameter ranges for list-decodable codes and hardness amplification. In list-decodable codes, one typically wants the agreement parameter ε to be a constant and the codeword length to be linear in the message length (i.e. $D \log q = O(n)$). In hardness amplification, ε is usually taken to be vanishingly small (even as small as $1/n^{\Omega(1)}$), and one can usually afford for the codeword length to be polynomial in the message length (i.e. $D \log q = \mathrm{poly}(n)$), because this corresponds to a linear blow-up in the input length of the amplified function Amp^f as compared to f. Another difference is that in locally list-decodable codes, it is most natural to for the list size K to be comparable to the running time t of the decoder, so the decoder has time to enumerate the elements of the list. For hardness amplification against nonuniform circuits, we may as well allow for the number of advice bits k to be as large as the running time t, which means that the list size $K = 2^k$ can be exponential in t.

The fact that locally list-decodable codes imply worst-case-to-average-case hardness amplification was shown by Sudan et al. [STV]. The fact that black-box amplifications imply list-decodable codes was implicit in [Tre1], and was made explicit in [TV].

8 Pseudorandom Generators

A pseudorandom generator is a deterministic function that stretches a short seed of truly random bits into a long string of "pseudorandom" bits that look random to any efficient algorithm. The idea of bits "looking random" is formalized by the notion of computational indistinguishability, which is a computational analogue of statistical difference (cf., Definition 9).

Definition 16 ([GM]) *Random variables X and Y are (s, ε)* indistinguishable *if for every boolean circuit T of size s, we have*

$$\Pr[T(X) = 1] - \Pr[T(Y) = 1] \leq \varepsilon.$$

This is equivalent to the more standard definition in which we bound the absolute value of the left-hand side by replacing T with its complement (which does not affect standard measures of circuit size).

Now we can define a pseudorandom generator as a function stretching d truly random bits into $m > d$ bits that are computationally indistinguishable from m truly random bits.

Definition 17 ([BM, Yao]) *A function $G : [D] \to [M]$ is an (s, ε)* pseudorandom generator *if $G(U_{[D]})$ is (s, ε) indistinguishable from $U_{[M]}$.*

Pseudorandom generators are powerful tools for cryptography and for derandomization. See the surveys [Gol2, CRT, Kab]. As far as the parameters, we would like the seed length $d = \log D$ to be as small as possible relative to the output length $m = \log M$, and we typically want generators that fool circuits of size $s = \mathrm{poly}(m)$. The error parameter ε is usually not too important for derandomization (e.g. constant ε) suffices, but vanishing ε (e.g. $\varepsilon = 1/\mathrm{poly}(m)$) is typically achievable and is crucial for cryptographic applications.

Another important parameter is the complexity of computing the generator itself. Even though this will not be explicit below, our discussions refer to pseudorandom generators whose running time may be larger than the distinguishers T they fool, e.g. polynomial in s and exponential in the seed length $d = \log D$. The study of such generators was initiated by Nisan and Wigderson [NW]. They generators suffice for derandomization, where we allow a polynomial slowdown in the algorithm we derandomize and anyhow enumerate over all $D = 2^d$ seeds. They are not suitable, however, for most cryptographic applications, where the generator is run by the honest parties, and must fool adversaries that have much greater running time.

The advantage of "noncryptographic" pseudorandom generators, whose running time is greater than that of the distinguishers, is that they can be constructed under weaker assumptions. The existence of "cryptographic" generators is equivalent to the existence of one-way functions [HILL], whereas "noncryptographic" generators can be constructed from any boolean function (in $\mathbf{E} = \mathbf{DTIME}(2^{O(n)})$) with high circuit complexity [NW, BFNW].

We formalize the notion of a black-box construction of pseudorandom generators from functions of high worst-case circuit complexity analogously to Definition 13.

Definition 18 *Let $G^f : [D] \to [M]$ be an algorithm that is defined for every oracle $f : [n] \to \{0,1\}$. We say that G is a (t,k,ε)* black-box PRG construction *if there is an oracle algorithm* Red*, running in time t, such that for every $T : [M] \to \{0,1\}$ such that*

$$\Pr[T(G^f(U_{[D]})) = 1] - \Pr[T(U_{[M]}) = 1] > \varepsilon,$$

there is an advice string $z \in [K]$ such that

$$\forall x \qquad \mathrm{Red}^T(x,z) = f(x).$$

Analogously to Proposition 14, black-box constructions according to the above definition do suffice for constructing pseudorandom generators from functions of high circuit complexity.

Proposition 19 *If* Amp *is a (t,k,ε) black-box hardness amplifier and f is $(0,s)$ hard, then G^f is an $(s/\tilde{O}(t),\varepsilon)$ pseudorandom generator.*

Again, for the purposes of this proposition, k may as well be taken to be equal to t, and the values of interest range from the "high end" $k = t = n^{\Omega(1)}$ (applicable for functions f whose circuit complexity in the input length, $\log n$) to the "low end" $k = t = (\log n)^{\Omega(1)}$ (applicable for functions f of superpolynomial circuit complexity).

We place pseudorandom generators in our framework analogously to hardness amplifiers. Given $G^f : [D] \to [M]$ defined for every oracle $f : [n] \to \{0,1\}$, we can define $\Gamma : [N] \times [D] \to [M]$ by

$$\Gamma(f,y) = G^f(y), \tag{8}$$

where $N = 2^n$. Again, if we allow the reduction unbounded running time, then black-box pseudorandom generator constructions can be characterized exactly in our framework.

Proposition 20 *Let $G^f : [D] \to [M]$ be an algorithm defined for every oracle $f : [n] \to \{0,1\}$, and let $\Gamma : [N] \times [D] \to [M]$ be the function corresponding to G via (8). Then G is an (∞,k,ε) black-box hardness amplifier if and only if*

$$\forall T \subseteq [M] \qquad |\mathrm{LIST}_\Gamma(T, \mu(T)+\varepsilon)| \leq K,$$

where $K = 2^k$ and $\mu(T) = |T|/M$.

The proof of Proposition 20 is similar to Proposition 15, noting that we can view a function $T : [M] \to \{0,1\}$ as the characteristic function of a set $T \subseteq [M]$ and conversely.

Notice that the condition in Proposition 20 is identical to the ones in our characterizations of averaging samplers (Proposition 5) and randomness extractors (Proposition 11). Thus, black-box pseudorandom generator constructions with reductions of unbounded running time are equivalent to both averaging samplers and randomness extractors. Analogously to the discussion of hardness amplifiers, an *efficient* reduction corresponds to extractors and samplers with efficient "local decoding" procedures. Here the decoder is given oracle access to a statistical test T that is trying to distinguish the output of the extractor Ext from uniform (respectively, a function T that the sampler Smp is trying to approximate). It should be able to efficiently compute any desired bit of any source string (resp., sequence of coin tosses) $x = f$ for which T succeeds in distinguishing the output $\mathrm{Ext}(x, U_{[D]})$ from uniform (resp., $\mathrm{Smp}(x)$ fails to approximate the density of T), given some $k = \log K$ bits of advice depending on x. Even though achieving this additional local decoding property seems to only make constructing extractors more difficult, the perspective it provides has proved useful in constructing extractors, because it suggests an algorithmic approach to establishing the extractor property (namely, designing an appropriate reduction/decoder).

In terms of parameters, black-box PRG constructions are closer to extractors than samplers. In particular, the "high end" of PRG constructions has $k = t = n^{\Omega(1)}$, corresponding to extracting randomness from sources whose min-entropy is polynomially smaller than the length. Another difference is that, with pseudorandom generator constructions, one typically only looks for an output length m that it is polynomially related to $t = k$. This corresponds to extractors that extract $m = k^{\Omega(1)}$ bits out of the k bits of min-entropy in the source, but for extractors, achieving $m = \Omega(k)$ or even $m \approx k + d$ is of interest. The connection between pseudorandom generators and extractors described here was discovered and first exploited by Trevisan [Tre1].

References

[Alo] N. Alon. Eigenvalues and expanders. *Combinatorica*, 6(2):83–96, 1986. Theory of computing (Singer Island, Fla., 1984).

[BFNW] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP Has Subexponential Time Simulations Unless EXPTIME has Publishable Proofs. *Computational Complexity*, 3(4):307–318, 1993.

[BR] M. Bellare and J. Rompel. Randomness-Efficient Oblivious Sampling. In *35th Annual Symposium on Foundations of Computer Science*, pages 276–287, Santa Fe, New Mexico, 20–22 Nov. 1994. IEEE.

[BM] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing*, 13(4):850–864, Nov. 1984.

[CG1] B. Chor and O. Goldreich. Unbiased Bits from Sources of Weak Randomness and Probabilistic Communication Complexity. *SIAM Journal on Computing*, 17(2):230–261, Apr. 1988.

[CG2] B. Chor and O. Goldreich. On the Power of Two-Point Based Sampling. *Journal of Complexity*, 5(1):96–106, Mar. 1989.

[CRT] A. E. F. Clementi, J. D. P. Rolim, and L. Trevisan. Recent advances towards proving P=BPP. *Bulletin of the European Association for Theoretical Computer Science. EATCS*, 64:96–103, 1998.

[GT] D. Galvin and P. Tetali. Slow mixing of Glauber dynamics for the hard-core model on regular bipartite graphs. *Random Structures & Algorithms*, 28(4):427–443, 2006.

[Gil] D. Gillman. A Chernoff bound for random walks on expander graphs. *SIAM J. Comput.*, 27(4):1203–1220 (electronic), 1998.

[Gol1] O. Goldreich. A Sample of Samplers - A Computational Perspective on Sampling (survey). *Electronic Colloquium on Computational Complexity (ECCC)*, 4(20), 1997.

[Gol2] O. Goldreich. *Modern cryptography, probabilistic proofs and pseudorandomness*, volume 17 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 1999.

[GM] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, Apr. 1984.

[Gur] V. Guruswami. *Algorithmic Results in List Decoding*, volume 2, number 2 of *Foundations and Trends in Theoretical Computer Science*. now publishers, 2006.

[GUV] V. Guruswami, C. Umans, and S. Vadhan. Unbalanced Expanders and Randomness Extractors from Parvaresh–Vardy Codes. In *Proceedings of the 22nd Annual IEEE Conference on Computational Complexity (CCC '07)*, 12–16 June 2007.

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

[HLW] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin of the AMS*, 43(4):439–561, 2006.

[Kab] V. Kabanets. Derandomization: a brief overview. *Bulletin of the EATCS*, 76:88–103, 2002.

[Mil] P. Miltersen. *Handbook of Randomized Computing*, chapter Derandomizing Complexity Classes. Kluwer, 2001.

[NT] N. Nisan and A. Ta-Shma. Extracting Randomness: A Survey and New Constructions. *Journal of Computer and System Sciences*, 58(1):148–173, February 1999.

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

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

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

[SSS] J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. *SIAM Journal on Discrete Mathematics*, 8(2):223–250, 1995.

[Sha] R. Shaltiel. Recent Developments in Extractors. In G. Paun, G. Rozenberg, and A. Salomaa, editors, *Current Trends in Theoretical Computer Science*, volume 1: Algorithms and Complexity. World Scientific, 2004.

[Sud] M. Sudan. List decoding: Algorithms and applications. *SIGACT News*, 31(1):16–27, 2000.

[STV] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom Generators without the XOR Lemma. *Journal of Computer and System Sciences*, 62:236–266, 2001.

[TUZ] A. Ta-Shma, C. Umans, and D. Zuckerman. Loss-less condensers, unbalanced expanders, and extractors. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, pages 143–152 (electronic), New York, 2001. ACM.

[TZ] A. Ta-Shma and D. Zuckerman. Extractor codes. *IEEE Transactions on Information Theory*, 50(12):3015–3025, 2004.

[Tre1] L. Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879 (electronic), 2001.

[Tre2] L. Trevisan. Some Applications of Coding Theory in Computational Complexity. *Quaderni di Matematica*, 13:347–424, 2004.

[TV] L. Trevisan and S. Vadhan. Pseudorandomness and Average-Case Complexity via Uniform Reductions. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity (CCC '02)*, pages 129–138, Montréal, CA, May 2002. IEEE.

[Vad] S. Vadhan. Lecture Notes on Pseudorandomness. `http://eecs.harvard.edu/~salil/cs225`, Spring 2007.

[Yao] A. C. Yao. Theory and Applications of Trapdoor Functions (Extended Abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, 3–5 Nov. 1982. IEEE.

[Zuc1] D. Zuckerman. Simulating BPP Using a General Weak Random Source. *Algorithmica*, 16(4/5):367–391, Oct./Nov. 1996.

[Zuc2] D. Zuckerman. Randomness-Optimal Oblivious Sampling. *Random Structures & Algorithms*, 11(4):345–367, 1997.