Contents

5

Randomness Extractors

We now move on to the second major pseudorandom object of this survey: randomness extractors. We begin by discussing the original motivation for extractors, which was to simulate randomized algorithms with sources of biased and correlated bits. This motivation is still compelling, but extractors have taken on a much wider significance in the years since they were introduced. They have found numerous applications in theoretical computer science beyond this initial motivating one, in areas from cryptography to distributed algorithms to metric embeddings. More importantly from the perspective of this survey, they have played a major unifying role in the theory of pseudorandomness. Indeed, the links between the various pseudorandom objects we will study in this survey (expander graphs, randomness extractors, list-decodable codes, pseudorandom generators, samplers) were all discovered through work on extractors.

5.1 Motivation and Definition

5.1.1 Deterministic Extractors

Typically, when we design randomized algorithms or protocols, we assume that all algorithms/parties have access to sources of *perfect* randomness, i.e. bits that are unbiased and completely independent. However, when we implement these algorithms, the physical sources of randomness to which we have access may contain biases and correlations. For example, we may use low-order bits of the system clock, the user's mouse movements, or a noisy diode based on quantum effects. While these sources may have some randomness in them, the assumption that the source is perfect is a strong one, and thus it is of interest to try and relax it.

Ideally, what we would like is a compiler that takes any algorithm A that works correctly when fed perfectly random bits U_m, and produces a new algorithm A' that will work even if it is fed random bits $X \in \{0,1\}^n$ that come from a "weak" random source. For example, if A is a **BPP** algorithm, then we would like A' to also run in probabilistic polynomial time. One way to design such compilers is to design a *randomness extractor* $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}^m$ such that $\mathrm{Ext}(X)$ is distributed uniformly in $\{0,1\}^m$.

IID-Bit Sources (aka Von Neumann Sources). A simple version of this question was already considered by von Neumann. He looked at sources that consist of boolean random variables $X_1, X_2, \ldots, X_n \in \{0,1\}$ that are independent but biased. That is, for every i, $\Pr[X_i = 1] = \delta$ for some unknown δ. How can such a source be converted into a source of independent, unbiased bits? Von Neumann proposed the following extractor: Break all the variables in pairs and for each pair output 0 if the outcome was 01, 1 if the outcome was 10 , and skip the pair if the outcome was 00 or 11. This will yield an unbiased random bit after $1/\delta$ pairs on average.

Independent-Bit Sources. Lets now look at a bit more interesting class of sources in which all the variables are still independent but the bias is no longer the same. Specifically, for every i, $\Pr[X_i = 1] = \delta_i$ and $0 < \delta \le \delta_i \le 1 - \delta$. How can we deal with such a source?

Let's be more precise about the problems we are studying. A *source* on $\{0,1\}^n$ is simply a random variable X taking values in $\{0,1\}^n$. In each of the above examples, there is an implicit *class* of sources being studied. For example, $\mathrm{IndBits}_{n,\delta}$ is the class of sources X on $\{0,1\}^n$ where the bits X_i are independent and satisfy $\delta \le \Pr[X_i = 1] \le 1 - \delta$. We could define $\mathrm{IIDBits}_{n,\delta}$ to be the same with the further restriction that all of the X_i's are identically distributed, i.e. $\Pr[X_i = 1] = \Pr[X_j = 1]$ for all i, j, thereby capturing von Neumann sources.

Definition 5.1 (deterministic extractors). [1] Let $\mathcal{C}$ be a class of sources on $\{0,1\}^n$. An ε-*extractor* for $\mathcal{C}$ is a function $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}^m$ such that for every $X \in \mathcal{C}$, $\mathrm{Ext}(X)$ is "ε-close" to U_m.

Note that we want a *single* function Ext that works for all sources in the class. This captures the idea that we do not want to assume we know the exact distribution of the physical source we are using, but only that it comes from some class. For example, for $\mathrm{IndBits}_{n,\delta}$, we know that the bits are independent and none are too biased, but not the specific bias of each bit. Note also that we only allow the extractor *one* sample from the source X. If we want to allow multiple independent samples, then this should be modelled explicitly in our class of sources; ideally we would like to minimize the independence assumptions used.

We still need to define what we mean for the output to be ε-close to U_m.

Definition 5.2. For random variables X and Y taking values in $\mathcal{U}$, their *statistical difference* (also known as *variation distance*) is $\Delta(X,Y) = \max_{T \subseteq \mathcal{U}} |\Pr[X \in T] - \Pr[Y \in T]|$. We say that X and Y are ε-*close* if $\Delta(X,Y) \le \varepsilon$.

Intuitively, any event in X happens in Y with the same probability $\pm\varepsilon$. This is really the most natural measure of distance for probability distributions (much more so than the ℓ_2 distance we used in the study of random walks). In particular, it satisfies the following natural properties.

Lemma 5.3 (properties of statistical difference). Let $X, Y, Z, X_1, X_2, Y_1, Y_2$ be random variables taking values in a universe $\mathcal{U}$. Then,

[1] Such extractors are called *deterministic* or *seedless* to contrast with the probabilistic or *seeded* randomness extractors we will see later.

(1) $\Delta(X,Y) \geq 0$, with equality iff X and Y are identically distributed,
(2) $\Delta(X,Y) \leq 1$, with equality iff X and Y have disjoint supports,
(3) $\Delta(X,Y) = \Delta(Y,X)$,
(4) $\Delta(X,Z) \leq \Delta(X,Y) + \Delta(X,Z)$,
(5) for every function f, we have $\Delta(f(X), f(Y)) \leq \Delta(X,Y)$,
(6) $\Delta((X_1,X_2),(Y_1,Y_2)) \leq \Delta(X_1,Y_1) + \Delta(X_2,Y_2)$ if X_1 and X_2, as well as Y_1 and Y_2, are independent, and
(7) $\Delta(X,Y) = \frac{1}{2} \cdot |X - Y|_1$, where $|\cdot|_1$ is the ℓ_1 distance. (Thus, X is ε-close to Y iff we can transform X into Y by "shifting" at most an ε fraction of probability mass.)

We now observe that extractors according to this definition give us the "compilers" we want.

Proposition 5.4. Let $A(w;r)$ be a randomized algorithm such that $A(w;U_m)$ has error probability at most γ, and let $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}^m$ be an ε-extractor for a class $\mathcal{C}$ of sources on $\{0,1\}^n$. Define $A'(w;x) = A(w;\mathrm{Ext}(x))$. Then for every source $X \in \mathcal{C}$, $A'(w;X)$ has error probability at most $\gamma + \varepsilon$.

This application identifies some additional properties we'd like from our extractors. We'd like the extractor itself to be efficiently computable (e.g. polynomial time). In particular, to get m almost-uniform bits out, we should need at most $n = \mathrm{poly}(m)$ bits bits from the weak random source.

We can cast our earlier extractor for sources of independent bits in this language:

Proposition 5.5. For every constant $\delta > 0$, every $n, m \in \mathbb{N}$, there is a polynomial-time computable function $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}^m$ that is an ε-extractor for $\mathrm{IndBits}_{n,\delta}$, with $\varepsilon = m \cdot 2^{-\Omega(n/m)}$.

In particular, taking $n = m^2$, we get exponentially small error with a source of polynomial length.

Proof. Ext breaks the source into m blocks of length $\lfloor n/m \rfloor$ and outputs the parity of each block. □

Unpredictable-Bit Sources (aka Santha–Vazirani Sources). Another interesting class of sources, which looks similar to the previous example is the class $\mathrm{UnpredBits}_{n,\delta}$ of *unpredictable-bit sources.* These are the sources that for every i, every $x_1, \ldots, x_n \in \{0,1\}$ and some constant $\delta > 0$, satisfy

$$\delta \leq \Pr\left[X_i = 1 \mid X_1 = x_1, X_2 = x_2, \ldots, X_{i-1} = x_{i-1}\right] \leq 1 - \delta$$

The parity extractor used above will be of no help with this source since the next bit could be chosen in a way that the parity will be equal to 1 with probability δ. Problem 5.4 shows that there does not exist any nontrivial extractor for these sources:

Proposition 5.6. For every $n \in \mathbb{N}$, $\delta > 0$, and fixed extraction function $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}$ there exists a source $X \in \mathrm{UnpredBits}_{n,\delta}$ such that either $\Pr[\mathrm{Ext}(X) = 1] \le \delta$ or $\Pr[\mathrm{Ext}(X) = 1] \ge 1 - \delta$. That is, there is no ε-extractor for $\mathrm{UnpredBits}_{n,\delta}$ for $\varepsilon < 1/2 - \delta$.

Nevertheless, as we will see, the answer to the question whether we can simulate **BPP** algorithms with unpredictable sources will be "yes"! Indeed, we will even be able to handle a much more general class of sources, introduced in the next section.

5.1.2 Entropy Measures and General Weak Sources

Intuitively, to extract m almost-uniform bits from a source, the source must have at least "m bits of randomness" in it (e.g. its support cannot be much smaller than 2^m). Ideally, this is all we would like to assume about a source. Thus, we need some measure of how much randomness is in a random variable; this can be done using various notions of *entropy* described below.

Definition 5.7 (entropy measures). Let X be a random variable. Then

- the *Shannon entropy* of X is:

$$\mathrm{H}_{Sh}(X) = \mathop{\mathrm{E}}_{x \overset{\mathrm{R}}{\leftarrow} X} \left[\log \frac{1}{\Pr[X = x]} \right].$$

- the *Rényi entropy* of X is:

$$\mathrm{H}_2(X) = \log \left(\frac{1}{\mathrm{E}_{x \overset{\mathrm{R}}{\leftarrow} X}[\Pr[X = x]]} \right) = \log \frac{1}{\mathrm{CP}(X)}, \text{ and}$$

- the *min-entropy* of X is:

$$\mathrm{H}_\infty(X) = \min_x \left\{ \log \frac{1}{\Pr[X = x]} \right\},$$

where all logs are base 2.

All the three measures satisfy the following properties we would expect from a measure of randomness:

Lemma 5.8 (properties of entropy). For each of of the entropy measures $\mathrm{H} \in \{\mathrm{H}_{Sh}, \mathrm{H}_2, \mathrm{H}_\infty\}$ and random variables X, Y, we have:

- $\mathrm{H}(X) \ge 0$, with equality iff X is supported on a single element,
- $\mathrm{H}(X) \le \log |\mathrm{Supp}(X)|$, with equality iff X is uniform on $\mathrm{Supp}(X)$,
- if X, Y are independent, then $\mathrm{H}((X,Y)) = \mathrm{H}(X) + \mathrm{H}(Y)$,
- for every deterministic function f, we have $\mathrm{H}(f(X)) \le \mathrm{H}(X)$, and
- for every X, we have $\mathrm{H}_\infty(X) \le \mathrm{H}_2(X) \le \mathrm{H}_{Sh}(X)$.

To illustrate the differences between the three notions, consider a source X such that $X = 0^n$ with probability 0.99 and $X = U_n$ with probability 0.01. Then $\mathrm{H}_{Sh}(X) \geq 0.01n$ (contribution from the uniform distribution), $\mathrm{H}_2(X) \leq \log(1/.99^2) < 1$ and $\mathrm{H}_\infty(X) \leq \log(1/.99) < 1$ (contribution from 0^n). Note that even though X has Shannon entropy linear in n, we cannot expect to extract bits that are close to uniform or carry out any useful randomized computations with one sample from X, because it gives us nothing useful 99% of the time. Thus, we should use the stronger measures of entropy given by H_2 or H_∞.

Then why is Shannon entropy so widely used in information theory results? The reason is that such results typically study what happens when you have many independent samples from the source. In such a case, it turns out that the the source is "close" to one where the min-entropy is roughly equal to the Shannon entropy. Thus the distinction between these entropy measures becomes less significant. (Recall that we only allow one sample from the source.) Moreover, Shannon entropy satisfies many nice identities that make it quite easy to work with. Min-entropy and Renyi entropy are much more delicate.

We will consider the task of extracting randomness from sources where all we know is a lower bound on the min-entropy:

Definition 5.9. X is a *k-source* is $\mathrm{H}_\infty(X) \geq k$, i.e., if $\Pr[X = x] \leq 2^{-k}$.

A typical setting of parameters is $k = \delta n$ for some fixed δ, e.g., 0.01. We call δ the *min-entropy rate*. Some different ranges that are commonly studied (and are useful for different applications): $k = \mathrm{polylog}(n)$, $k = n^\gamma$ for a constant $\gamma \in (0, 1)$, $k = \delta n$ for a constant $\delta \in (0, 1)$, and $k = n - O(1)$. The middle two ($k = n^\gamma$ and $k = \delta n$) are the most natural for simulating randomized algorithms with weak random sources.

Examples of k-sources:

- k random and independent bits, together with $n - k$ fixed bits (in an arbitrary order). These are called *oblivious bit-fixing* sources.
- k random and independent bits, and $n-k$ bits that depend arbitrarily on the first k bits. These are called *adaptive bit-fixing sources.*
- Unpredictable-bit sources with bias parameter δ. These are k-sources with $k = \log(1/(1-\delta)^n) = \Theta(\delta n)$.
- Uniform distribution on $S \subset \{0, 1\}^n$ with $|S| = 2^k$. These are called *flat k-sources.*

It turns out that flat k-sources are really representative of general k-sources.

Lemma 5.10. Every k-source is a convex combination of flat k-sources (provided that $2^k \in \mathbb{N}$), i.e., $X = \sum p_i X_i$ with $0 \leq p_i \leq 1$, $\sum p_i = 1$ and all the X_i are flat k-sources.

Proof. [Sketch] Consider each source on $[N]$ (recall that $N = 2^n$) as a vector $X \in \mathbb{R}^N$. Then X is a k-source if and only if $X(i) \in [0, 2^{-k}]$ for every $i \in [N]$ and $\sum_i X(i) = 1$. The set of vectors X satisfying these linear inequalities is a polytope. By basic linear programming theory, all of the points in the polytope are convex combinations of its *vertices*, which are defined to be the points that make a maximal subset of the inequalities tight. By inspection, the vertices of the polytope of k-sources are those sources where $X(i) = 2^{-k}$ for 2^k values of i and $X(i) = 0$ for the remaining values of i; these are simply the k-sources. $\square$

By Lemma 5.10, we can think of any k-source as being obtained by first selecting a flat k-source X_i according to some distribution (given by the p_i's) and then selecting a random sample from X_i. This means that if we can compile probabilistic algorithms to work with flat k-sources, then we can compile them to work with any k-source.

5.1.3 Seeded Extractors

Proposition 5.6 tell us that it impossible to have deterministic extractors for unpredictable sources. Here we consider k-sources, which are more general than unpredictable sources, and hence it is also impossible to have deterministic extractors for them. The impossibility result for k-sources is stronger and simpler to prove.

Proposition 5.11. For any $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}$ there exists an $(n-1)$-source X so that $\mathrm{Ext}(X)$ is constant.

Proof. There exists $b \in \{0,1\}$ so that $|\mathrm{Ext}^{-1}(b)| \geq 2^n/2 = 2^{n-1}$. Then let X be the uniform distribution on $\mathrm{Ext}^{-1}(b)$. $\square$

On the other hand, if we reverse the order of quantifiers, allowing the extractor to depend on the source, it is easy to see that good extractors exist and in fact a randomly chosen function will be a good extractor with high probability.

Proposition 5.12. For every $n, k, m \in \mathbb{N}$, every $\varepsilon > 0$, and every flat k-source X, if we choose a random function $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}^m$ with $m = k - 2\log(1/\varepsilon) - O(1)$, then $\mathrm{Ext}(X)$ will be ε-close to U_m with probability $1 - 2^{-\Omega(K\varepsilon^2)}$, where $K = 2^k$.

(We will commonly use the convention that capital variables are 2 raised to the power of the corresponding lowercase variable, such as $K = 2^k$ above.)

Proof. Choose our extractor randomly. We want it to have following property: for all $T \subseteq [M]$, $|\Pr[\mathrm{Ext}(X) \subseteq T] - \Pr[U_m \subseteq T| \leq \varepsilon$. Equivalently, $|\{x \in \mathrm{Supp}(X) : \mathrm{Ext}(x) \in T\}|/K$ differs from the density $\mu(T)$ by at most ε. For each point $x \in \mathrm{Supp}(X)$, the probability that $\mathrm{Ext}(x) \in T$ is $\mu(T)$, and these events are independent. By the Chernoff Bound, for each fixed T, this condition holds with probability at least $1 - 2^{-\Omega(K\varepsilon^2)}$. Then the probability that condition is violated for at least one T is at most $2^M 2^{-\Omega(K\varepsilon^2)}$, which is less than 1 for $m = k - 2\log(1/\varepsilon) - O(1)$. $\square$

Note that the failure probability is doubly-exponentially small in k. Naively, one might hope that we could get an extractor that's good for all flat k-sources by a union bound. But the number of flat k-sources is $\binom{N}{K} \approx N^K$ (where $N = 2^n$), which is unfortunately a larger double-exponential in k. We can overcome this gap by allowing the extractor to be "slightly" probabilistic, i.e. allowing the extractor a *seed* consisting of a small number of truly random bits in addition to the weak random source. We can think of this seed of truly random bits as a random choice of an extractor from family of extractors. This leads to the following crucial definition:

Definition 5.13 (seeded extractors). Extractor $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ε)-*extractor* if for every k-source X on $\{0,1\}^n$, $\mathrm{Ext}(X, U_d)$ is ε-close to U_m.

We want to give a construction that minimizes d and maximizes m. We prove the following theorem.

Theorem 5.14. For every $n \in \mathbb{N}$, $k \in [0, n]$ and $\varepsilon > 0$, there exists a (k, ε)-extractor $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with $m = k + d - 2\log(1/\varepsilon) - O(1)$ and $d = \log(n-k) + 2\log(1/\varepsilon) + O(1)$.

One setting of parameters to keep in mind (for our application of simulating randomized algorithms with a weak source) is $k = \delta n$, with δ a fixed constant (e.g. $\delta = 0.01$), and ε a fixed constant (e.g. $\varepsilon = 0.01$).

Proof. We use the Probabilistic Method. By Lemma 5.10, it suffices for Ext to work for flat k-sources. Choose the extractor Ext at random. Then the probability that the extractor fails is not more than number of flat k-sources times times the probability Ext fails for a fixed flat k-source. By the above proposition, the probability of failure for a fixed flat k-source is at most $2^{-\Omega(KD\varepsilon^2)}$, since (X, U_d) is a flat $(k+d)$-source) and $m = k + d - 2\log(\frac{1}{\varepsilon}) - O(1)$. Thus the total failure probability is at most

$$\binom{N}{K} \cdot 2^{-\Omega(KD\varepsilon^2)} \le \left(\frac{Ne}{K}\right)^K 2^{-\Omega(KD\varepsilon^2)}.$$

The latter expression is less than 1 if $D\varepsilon^2 \ge c\log(Ne/K) = c \cdot (n-k) + c'$ for constants c, c'.This is equivalent to $d = \log(n-k) + 2\log(\frac{1}{\varepsilon}) + O(1)$. □

It turns out that both bounds (on m and d) are individually tight up to the $O(1)$ terms.

Recall that our motivation for extractors was to simulate randomized algorithms given *only* a weak random source, so allowing a truly random seed may seem to defeat the purpose. However, if the seed is of logarithmic length as in Theorem 5.14, then instead of selecting it randomly, we can enumerate all possibilities for the seed and take a majority vote.

Proposition 5.15. Let $A(w; r)$ be a randomized algorithm such that $A(w; U_m)$ has error probability at most γ, and let $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be a (k, ε)-extractor. Define

$$A'(w; x) = \operatorname*{maj}_{y \in \{0,1\}^d} \{A(w; \mathrm{Ext}(x, y))\}.$$

Then for every k-source X on $\{0,1\}^n$, $A'(w; X)$ has error probability at most $2(\gamma + \varepsilon)$.

Proof. The probability that $A(w; \mathrm{Ext}(X, U_d))$ is incorrect is not more than probability $A(w; U_m)$ is incorrect plus ε, i.e. $\gamma + \varepsilon$, by the definition of statistical difference. Then the probability that $\mathrm{maj}_y\, A(w, \mathrm{Ext}(X, y))$ is incorrect is at most $2 \cdot (\gamma + \varepsilon)$, because each error of $\mathrm{maj}_y\, A(w; \mathrm{Ext}(x, y))$ corresponds to $A(w; \mathrm{Ext}(x, U_d))$ erring with probability at least $1/2$. □

Note that the enumeration incurs a 2^d factor slowdown in the simulation. Thus, for this application, we want to construct extractors where (a) $d = O(\log n)$; (b) Ext is computable in polynomial time; and (c) $m = k^{\Omega(1)}$.

We remark that the error probability in Proposition 5.15 can actually be made exponentially small (say 2^{-t}) by using an extractor that is designed for min-entropy roughly $k - t$ instead of k.

We note that even though seeded extractors suffice for simulating randomized algorithms with only a weak source, they do not suffice for all applications of randomness in theoretical computer science. The trick of eliminating the random seed by enumeration does not work, for example, in cryptographic applications of randomness. Thus the study of deterministic extractors for restricted classes of sources remains a very interesting and active research direction. We, however, will focus on seeded extractors, due to their many applications and their connections to the other pseudorandom objects we are studying.

5.2 Connections with Hash Functions and Expanders

As mentioned earlier, extractors have played a unifying role in the theory of pseudorandomness, through their close connections with a variety of other pseudorandom objects. In this section, we will see two of these connections. Specifically, how by reinterpreting them appropriately, extractors can be viewed as providing families of hash functions, and as being a certain type of highly expanding graphs.

5.2.1 Extractors as Hash Functions

One of the results we saw last time says that for any subset $S \subseteq [N]$ of size K, if we choose a completely random hash function $h : [N] \to [M]$ for $M \ll K$, then h will map the elements of S almost-uniformly to $[M]$. Equivalently, if we let H be distributed uniformly over all functions $h : [N] \to [M]$ and X be uniform on the set S, then $(H, H(X))$ is statistically close to $(H, U_{[M]})$, where we use the notation U_T to denote the uniform distribution on a set T. Can we use a smaller family of hash functions than the set of all functions $h : [N] \to [M]$? This gives rise to the following variant of extractors.

Definition 5.16 (strong extractors). Extractor $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a *strong* (k,ε)*-extractor* if for every k-source X on $\{0,1\}^n$, $(U_d, \mathrm{Ext}(X, U_d))$ is ε-close to (U_d, U_m). Equivalently, $\mathrm{Ext}'(x,y) = (y, \mathrm{Ext}(x,y))$ is a standard (k,ε)-extractor.

The nonconstructive existence proof from last time can be extended to establish the existence of very good strong extractors.

Theorem 5.17. For every $n, k \in \mathbb{N}$ and $\varepsilon > 0$ there exists a strong (k,ε)-extractor $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with $m = k - 2\log(\frac{1}{\varepsilon}) - O(1)$ and $d = \log(n-k) + 2\log(\frac{1}{\varepsilon}) + O(1)$.

Note that the output length $m \approx k$ instead of $m \approx k + d$; intuitively a strong extractor needs to extract randomness that is *independent* of the seed and thus can only get the k bits from the source.

We see that strong extractors can be viewed as very small families of hash functions having the almost-uniform mapping property mentioned above. Indeed, our first explicit construction of extractors is obtained by using pairwise independent hash functions.

Theorem 5.18 (Leftover Hash Lemma). If $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ is a pairwise independent (or even 2-universal) family of hash functions where $m = k - 2\log(\frac{1}{\varepsilon})$, then $\mathrm{Ext}(x,h) \overset{\text{def}}{=} h(x)$ is a strong (k,ε)-extractor. Equivalently, $\mathrm{Ext}(x,h) = (h, h(x))$ is a standard (k,ε)-extractor.

Note that the seed length is $d = O(n)$, i.e., the number of random bits required to choose $h \overset{\text{R}}{\leftarrow} \mathcal{H}$. This is far from optimal; for the purposes of simulating randomized algorithms we would like $d = O(\log n)$. However, the output length of the extractor is $m = k - 2\log(\frac{1}{\varepsilon})$, which is optimal up to an additive constant.

Proof. Let X be an arbitrary k-source on $\{0,1\}^n$, $\mathcal{H}$ as above, and $H \overset{\text{R}}{\leftarrow} \mathcal{H}$. Let d be the seed length. We show that $(H, H(X))$ is ε-close to $U_d \times U_m$ in the following three steps:

(1) We show that the collision probability of $(H, H(X))$ is close to that of $U_d \times U_m$.
(2) We note that this is equivalent to saying that the ℓ_2 distance between $(H, H(X))$ and $U_d \times U_m$ is small.
(3) Then we deduce that the statistical difference is small, by recalling that the statistical difference equals half of the ℓ_1 distance, which can be (loosely) bounded by the ℓ_2 distance.

Proof of 1: By definition, $\mathrm{CP}(H, H(X)) = \Pr[(H, H(X)) = (H', H'(X'))]$, where (H', X') is independent of and identically distributed to (H, X). Note that $(H, H(X)) = (H', H'(X))$ if and only if $H = H'$ and either $X = X'$ or $X \neq X'$ but $H(X) = H(X')$. Thus

$$\begin{aligned}\mathrm{CP}(H, H(X)) &= \mathrm{CP}(H) \cdot \Big(\mathrm{CP}(X) + \Pr\left[H(X) = H(X') \mid X \neq X'\right]\Big) \\ &\leq \frac{1}{D}\left(\frac{1}{K} + \frac{1}{M}\right) = \frac{1+\varepsilon^2}{DM}.\end{aligned}$$

To see the penultimate inequality, note that $\mathrm{CP}(H) = 1/D$ because there are D hash functions, $\mathrm{CP}(X) \leq 1/K$ because $\mathrm{H}_\infty(X) \geq k$, and $\Pr[H(X) = H(X') \,| X \neq X'] = 1/M$ by pairwise independence.

Proof of 2:

$$\begin{aligned}\|(H, H(X)) - U_d \times U_m\|^2 &= \mathrm{CP}(H, H(X)) - \frac{1}{DM} \\ &\leq \frac{1+\varepsilon^2}{DM} - \frac{1}{DM} = \frac{\varepsilon^2}{DM}.\end{aligned}$$

Proof of 3: Recalling that the statistical difference between two random variables X and Y is equal to $\frac{1}{2}|X-Y|_1$, we have:

$$\begin{aligned}\Delta((H,H(X)),U_d\times U_m) &= \frac{1}{2}\,|(H,H(X))-U_d\times U_m|_1\\ &\le \frac{\sqrt{DM}}{2}\,\|(H,H(X))-U_d\times U_m\|\\ &\le \frac{\sqrt{DM}}{2}\cdot\sqrt{\frac{\varepsilon^2}{DM}}\\ &= \frac{\varepsilon}{2}.\end{aligned}$$

Thus, we have in fact obtained a strong $(k,\varepsilon/2)$-extractor. □

The proof above actually shows that $\mathrm{Ext}(x,h)=h(x)$ extracts with respect to collision probability, or equivalently, with respect to the ℓ_2-norm. This property may be expressed in terms of Renyi entropy $\mathrm{H}_2(Z)\overset{\text{def}}{=}\log(1/\mathrm{CP}(Z))$. Indeed, we can define $\mathrm{Ext}:\{0,1\}^n\times\{0,1\}^d\longrightarrow\{0,1\}^m$ to be a (k,ε) *Renyi-entropy extractor* if $\mathrm{H}_2(X)\ge k$ implies $\mathrm{H}_2(\mathrm{Ext}(X,U_d))\ge m-\varepsilon$ (or $\mathrm{H}_2(U_d,\mathrm{Ext}(X,U_d))\ge m+d-\varepsilon$ for strong Renyi-entropy extractors). Then the above proof shows that pairwise-independent hash functions yield strong Renyi-entropy extractors.

In general, it turns out that an extractor with respect to Renyi entropy must have seed length $d\ge\Omega(\min\{m,n-k\})$ (as opposed to $d=O(\log n)$); this explains why the seed length in the above extractor is large.

5.2.2 Extractors vs. Expanders

Extractors have a natural interpretation as graphs. Specifically, we can interpret an extractor $\mathrm{Ext}:\{0,1\}^n\times\{0,1\}^d\longrightarrow\{0,1\}^m$ as the neighbor function of a bipartite multigraph $G=([N],[M],E)$ with $N=2^n$ left-vertices, $M=2^m$ right-vertices, and left-degree $D=2^d$,[2] where the r'th neighbor of left-vertex u is $\mathrm{Ext}(u,r)$. Typically $n\gg m$, so the graph is very unbalanced. It turns out that the extraction property of Ext is related to various "expansion" properties of G. In this section, we explore this relationship.

Let $\mathrm{Ext}:\{0,1\}^n\times\{0,1\}^d\longrightarrow\{0,1\}^m$ be a (k,ε)-extractor and $G=([N],[M],E)$ the associated graph. Recall that it suffices to examine Ext with respect to *flat* k-sources: in this case, the extractor property says that given a subset S of size k on the left, a random neighbor of a random element of S should be close to uniform on the right. In particular, if $S\subseteq[N]$ is a subset on the left of size k, then $|N(S)|\ge(1-\varepsilon)M$. This property is just like vertex expansion, except that it ensures expansion only for sets of size exactly K, not any size $\le K$. We call such a graph an $(=K,A)$ *vertex expander*. Indeed, this gives rise to the following weaker variant of extractors.

Definition 5.19 (dispersers). A function $\mathrm{Disp}:\{0,1\}^n\times\{0,1\}^d\to\{0,1\}^m$ is a (k,ε)-*disperser* if for every k-source X on $\{0,1\}^n$, $\mathrm{Disp}(X,U_d)$ has a support of size at least $(1-\varepsilon)\cdot 2^m$.

While extractors can be used to simulate **BPP** algorithms with a weak random source, dispersers can be used to simulate **RP** algorithms with a weak random source.

Then, we have:

[2] This connection is the reason we use d to denote the seed length of an extractor.

Proposition 5.20. A function $\mathrm{Disp} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ε)-disperser iff the corresponding bipartite multigraph $G = ([N], [M], E)$ with left-degree D is an $(= K, A)$ vertex expander for $A = (1 - \varepsilon) \cdot M/K$.

Note that extractors and dispersers are interesting even when $M \ll K$, so the expansion parameter A may be less than 1. Indeed, $A < 1$ is interesting for vertex "expanders" when the graph is highly imbalanced. Still, for an *optimal* extractor, we have $M = \Theta(\varepsilon^2 KD)$ (because $m = k + d - 2\log(1/\varepsilon) - \Theta(1)$), which corresponds to expansion factor $A = \Theta(\varepsilon^2 D)$. (An optimal disperser actually gives $A = \Theta(D/\log(1/\varepsilon))$.) Note this is smaller than the expansion factor of $D/2$ in Ramanujan graphs and $D - O(1)$ in random graphs; the reason is that those expansion factors are for "small" sets, whereas here we are asking for sets to expand to almost the entire right-hand side.

Now let's look for a graph-theoretic property that is *equivalent* to the extraction property. Ext is a (k, ε)-extractor iff for every set $S \subseteq [N]$ of size K,

$$\Delta(\mathrm{Ext}(U_S, U_{[D]}), U_{[M]}) = \max_{T \subseteq [M]} \left| \Pr\left[\mathrm{Ext}(U_S, U_{[D]}) \in T\right] - \Pr\left[U_{[M]} \in T\right] \right| \le \varepsilon,$$

where U_S denotes the uniform distribution on S. This inequality may be expressed in graph-theoretic terms as follows. For every set $T \subseteq [M]$,

$$\begin{aligned} & \left| \Pr\left[\mathrm{Ext}(U_S, U_{[D]}) \in T\right] - \Pr\left[U_{[M]} \in T\right] \right| \le \varepsilon \\ \Leftrightarrow \quad & \left| \frac{e(S,T)}{|S|D} - \frac{|T|}{M} \right| \le \varepsilon \\ \Leftrightarrow \quad & \left| \frac{e(S,T)}{ND} - \mu(S)\mu(T) \right| \le \varepsilon\mu(S), \end{aligned}$$

where $e(S, T)$ is the number of edges from S to T.

Thus, we have:

Proposition 5.21. Ext is a (k, ε)-extractor iff the corresponding bipartite multigraph $G = ([N], [M], E)$ with left-degree D has the property that $|e(S,T)/ND - \mu(S)\mu(T)| \le \varepsilon\mu(S)$ for every $S \subseteq [N]$ of size K and every $T \subseteq [M]$.

Note that this is very similar to the Expander Mixing Lemma (Lemma 4.15) which states that if a graph G has spectral expansion λ, then for *all* sets $S, T \subseteq [N]$ we have

$$\left| \frac{e(S,T)}{ND} - \mu(S)\mu(T) \right| \le \lambda\sqrt{\mu(S)\mu(T)}.$$

It follows that if $\lambda\sqrt{\mu(S)\mu(T)} \le \varepsilon\mu(S)$ for all $S \subseteq [N]$ of size K and all $T \subseteq [N]$, then G gives rise to a (k, ε)-extractor (by turning G into a D-regular bipartite graph with N vertices on each side in the natural way). It suffices for $\lambda \le \varepsilon \cdot \sqrt{K/N}$ for this to work.

We can use this connection to turn our explicit construction of spectral expanders into an explicit construction of extractors. To achieve $\lambda \le \varepsilon \cdot \sqrt{K/N}$, we can take an appropriate power of a constant-degree expander. Specifically, if G_0 is a D_0-regular expander on N vertices with bounded second eigenvalue, we can consider the tth power of G_0, $G = G_0^t$, where $t = O(\log((1/\varepsilon)\sqrt{N/K})) = O(n-k+\log(1/\varepsilon))$. The degree of G is $D = D_0^t$, so $d = \log D = O(t)$. This yields the following result:

Theorem 5.22. For every $n, k \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit (k, ε)-extractor $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \longrightarrow \{0,1\}^n$ with $d = O(n-k+\log(1/\varepsilon))$.

Note that the seed length is significantly better than in the construction from pairwise-independent hashing when k is close to n, say $k \geq n - O(\log n)$ (i.e. $K = \Omega(N/\log N)$). The output length is just n, which is much larger than the typical output length for extractors (usually $m \ll n$). Using a Ramanujan graph (rather than an arbitrary constant-degree expander), the seed length can be improved to $d = n - k + 2\log(1/\varepsilon) + O(1)$, which yields an optimal output length $n = k + d - 2\log(1/\varepsilon) - O(1)$.

Another way of proving Theorem 5.22 is to use the fact that a random step on an expanders decreases the ℓ_2 distance to uniform, like in the proof of the Leftover Hash Lemma. This analysis shows that we actually get a Renyi-entropy extractor; and thus explains the large seed length $d \approx n - k$.

The following table summarizes the main differences between "classic" expanders and extractors.

Expanders	**Extractors**
Measured by vertex or spectral expansion	Measured by min-entropy/statistical difference
Typically constant degree	Typically logarithmic or poly-logarithmic degree
All sets of size *at most* K expand	All sets of size *exactly* (or at least) K expand
Typically balanced	Typically unbalanced, bipartite graphs

Fig. 5.1 Differences between "classic" expanders and extractors

5.3 Constructing Extractors

In the previous sections, we have seen that very good extractors exist — extracting almost all of the min-entropy from a source with only a logarithmic seed length. But the explicit constructions we have seen (via pairwise-independent hashing and spectral expanders) are still quite far from optimal in seed length, and in particular cannot be used to give a polynomial-time simulation of **BPP** with a weak random source.

Fortunately, much better extractor constructions are known — ones that extract any constant fraction of the min-entropy using a logarithmic seed length, or extract all of the min-entropy using a polylogarithmic seed length. In this lecture, we will see how to construct such extractors (assuming a certain *condenser* construction that we will see in Chapter 6).

5.3.1 Block Sources

We introduce a useful model of sources that has more structure than an arbitrary k-source:

Definition 5.23. $X = (X_1, X_2, \ldots, X_t)$ is a $(k_1, k_2, \ldots, k_t)$ *block source* if for every $x_1, \ldots, x_{i-1}$, $X_i|_{X_1=x_1,\ldots,X_{i-1}=x_{i-1}}$ is a k_i-source. If $k_1 = k_2 = \cdots = k_t = k$, then we call X a $t \times k$ *block source.*

Note that a $(k_1, k_2, \ldots, k_t)$ block source is also a $(k_1 + \cdots + k_t)$-source, but it comes with additional structure — each block is guaranteed to contribute some min-entropy. Thus, extracting randomness from block sources is an easier task than extracting from general sources.

The study of block sources has a couple of motivations.

- They are a natural and plausible model of sources in their own right. Indeed, they are more general than unpredictable-bit sources of Section 5.1.1: if $X \in \mathrm{UnpredBits}_{n,\delta}$ is broken into t blocks of length $\ell = n/t$, then the result is a $t \times \delta'\ell$ block source, where $\delta' = \log(1/(1-\delta))$.
- We can construct extractors for general weak sources by converting a general weak source into a block source. We will see how to do this later in the lecture.

We now illustrate how extracting from block sources is easier than from general sources. The idea is that we can extract almost-uniform bits from later blocks that are essentially independent of earlier blocks, and hence use these as a seed to extract more bits from the earlier blocks. Specifically, for the case of two blocks we have the following:

Lemma 5.24. Let $\mathrm{Ext}_1 : \{0,1\}^{n_1} \times \{0,1\}^{d_1} \to \{0,1\}^{m_1}$ be a (k_1, ε_1)-extractor, and $\mathrm{Ext}_2 : \{0,1\}^{n_2} \times \{0,1\}^{d_2} \to \{0,1\}^{m_2}$ be a (k_2, ε_2)-extractor with $m_2 \geq d_1$. Define $\mathrm{Ext}'((x_1, x_2), y_2) = (\mathrm{Ext}_1(x_1, y_1), z_2)$, where (y_1, z_2) is obtained by partitioning $\mathrm{Ext}_2(x_2, y_2)$ into a prefix y_1 of length d_1 and a suffix z_2 of length $m_2 - d_1$.

Then for every (k_1, k_2) block source $X = (X_1, X_2)$ taking values in $\{0,1\}^{n_1} \times \{0,1\}^{n_2}$, it holds that $\mathrm{Ext}'(X, U_{d_2})$ is $(\varepsilon_1 + \varepsilon_2)$-close to $U_{m_1} \times U_{m_2-d_1}$.

Proof. Since X_2 is a k_2-source conditioned on any value of X_1 and Ext_2 is a (k_2, ε_2)-extractor, it follows that $(X_1, Y_1, Z_2) = (X_1, \mathrm{Ext}_2(X_2, U_{d_2}))$ is ε_2-close to $(X_1, U_{m_2}) = (X_1, U_{d_1}, U_{m_2-d_1})$.

Thus, $(\mathrm{Ext}_1(X_1, Y_1), Z_2)$ is ε_2-close to $(\mathrm{Ext}_1(X_1, U_{d_1}), U_{m_2-d_1})$, which is ε_1-close to $(U_{m_1}, U_{m_2-d_1})$ because X_1 is a k_1-source and Ext_1 is a (k_1, ε_1)-extractor.

By the triangle inequality, $\mathrm{Ext}'(X, U_{d_2}) = (\mathrm{Ext}_1(X_1, Y_1), Z_2)$ is $(\varepsilon_1 + \varepsilon_2)$-close to $(U_{m_1}, U_{m_2-d_1})$. □

The benefit of this composition is that the seed length of Ext' depends only one of the extractors (namely Ext_2) rather than being the sum of the seed lengths. (If this is reminiscent of the zig-zag product, it is because they are closely related — see Section 5.4.2). Thus, we get to extract from multiple blocks at the "price of one." Moreover, since we can take $d_1 = m_2$, which is typically much larger than d_2, the seed length of Ext' can even be smaller than that of Ext_1.

The lemma extends naturally to extracting from many blocks:

Lemma 5.25. For $i = 1, \ldots, t$, let $\mathrm{Ext}_i : \{0,1\}^{n_i} \times \{0,1\}^{d_i} \to \{0,1\}^{m_i}$ be a (k_i, ε_i)-extractor, and suppose that $m_i \geq d_{i-1}$ for every $i = 1, \ldots, t$, where we define $d_0 = 0$. Define $\mathrm{Ext}'((x_1, \ldots, x_t), y_t) =$

$(z_1, \ldots, z_t)$, where for $i = t, \ldots, 1$, we inductively define (y_{i-1}, z_i) to be a partition of $\mathrm{Ext}_i(x_i, y_i)$ into a d_{i-1}-bit prefix and a $(m_i - d_{i-1})$-bit suffix.

Then for every $(k_1, \ldots, k_t)$ block source $X = (X_1, \ldots, X_t)$ taking values in $\{0,1\}^{n_1} \times \cdots \{0,1\}^{n_t}$, it holds that $\mathrm{Ext}'(X, U_{d_t})$ is ε-close to U_m for $\varepsilon = \sum_{i=1}^t \varepsilon_i$ and $m = \sum_{i=1}^t (m_i - d_{i-1})$.

We remark that this composition preserves "strongness." If each of the Ext_i's correspond to strong extractors in the sense that their seeds are prefixes of their outputs, then Ext' will also correspond to a strong extractor. If in addition $d_1 = d_2 = \cdots = d_t$, then this construction can be seen as simply using the same seed to extract from all blocks.

Already with this simple composition, we can simulate **BPP** with an unpredictable-bit source (even though deterministic extraction from such sources is impossible by Proposition 5.6). As noted above, by breaking an unpredictable-bit source X with parameter δ into blocks of length ℓ, we obtain a $t \times k$ block source for $t = n/\ell$, $k = \delta'\ell$, and $\delta' = \log(1/(1-\delta))$.

Suppose that δ is a constant. Set $\ell = (10/\delta')\log n$, so that X is a $t \times k$ block source for $k = 10 \log n$, and define $\varepsilon = n^{-2}$. Letting $\mathrm{Ext} : \{0,1\}^\ell \times \{0,1\}^d \to \{0,1\}^{d+m}$ be the (k, ε) extractor using pairwise-independent hash functions (Theorem 5.18), we have:

$$\begin{aligned} d &= O(\ell) = O(\log n) \qquad \text{and} \\ m &= k - 2\log\frac{1}{\varepsilon} - O(1) > k/2 \end{aligned}$$

Composing Ext with itself t times as in Lemma 5.24, we obtain $\mathrm{Ext}' : \{0,1\}^{t\cdot\ell} \times \{0,1\}^d \to \{0,1\}^{d+t\cdot m}$ such that $\mathrm{Ext}'(X, U_d)$ is ε'-close to uniform, for $\varepsilon' = 1/n$. (Specifically, $\mathrm{Ext}'((x_1, \ldots, x_t), h) = (h, h(x_1), \ldots, h(x_t))$.) This tells us that Ext' essentially extracts half of the min-entropy from X, given a random seed of logarithmic length. Plugging this extractor into the construction of Proposition 5.15 gives us the following result.

Theorem 5.26. For every constant $\delta > 0$, we can simulate **BPP** with an unpredictable-bit source of parameter δ. More precisely, for every $L \in \mathbf{BPP}$ and every constant $\delta > 0$, there is a polynomial-time algorithm A and a polynomial q such that for every $w \in \{0,1\}^*$ and every source $X \in \mathrm{UnpredBits}_{q(|w|),\delta}$, the probability that $A(w; X)$ errs is at most $1/|w|$.

5.3.2 Reducing General Sources to Block Sources

Given the results of the previous section, a common approach to constructing extractors for general k-sources is to reduce the case of general k-sources to that of block sources.

One approach to doing this is as follows. Given a k-source X of length n, where $k = \delta n$, pick a (pseudo)random subset S of the bits of X, and let $W = X|_S$ be the bits of X in those positions. If the set S is of size ℓ, then we expect that W will have at least roughly $\delta\ell$ bits of min-entropy (with high probability over the choice of S). Moreover, W can have at most ℓ bits of min-entropy, so if $\ell < \delta n$, intuitively there must still should be at least min-entropy left in X. (This is justified by Lemma 5.27 below.) Thus, the pair (W, X) should be a block source. This approach can be shown to work for appropriate ways of sampling the set S, and recursive applications of it was the original approach to constructing good extractors (and is still useful in various contexts today). The fact mentioned above, that conditioning on a string of length ℓ reduces min-entropy by at most ℓ bits, is given by the following lemma (which is very useful when working with min-entropy) .

Lemma 5.27 (chain rule for min-entropy). If (W, X) are two jointly distributed random variables, where (W, X) is a k-source and W has length at most ℓ, then for every $\varepsilon > 0$, it holds that with probability at least $1 - \varepsilon$ over $w \overset{\mathrm{R}}{\leftarrow} W$, $X|_{W=w}$ is a $(k - \ell - \log(1/\varepsilon))$-source.

This is referred to as the "chain rule" for min-entropy by analogy with the chain rule for Shannon entropy, which states that $\mathrm{H}_{Sh}(X|W) = \mathrm{H}_{Sh}(W, X) - \mathrm{H}_{Sh}(W)$, where the conditional Shannon entropy is defined to be $\mathrm{H}_{Sh}(X|W) = \mathrm{E}_{w \overset{\mathrm{R}}{\leftarrow} W}[\mathrm{H}_{Sh}(X|_{W=w})]$. Thus, if $\mathrm{H}_{Sh}(W, X) \geq k$ and W is of length at most ℓ, we have $\mathrm{H}_{Sh}(X|W) \geq k - \ell$. The chain rule for min-entropy is not quite as clean; we need to assume that W has small support (rather than just small min-entropy) and we lose $\log(1/\varepsilon)$ bits of additional min-entropy.

Another approach, which we will follow, is based on the observation that every source of high min-entropy rate (namely, greater than 1/2) is (close to) a block source, as shown by the lemma below. Thus, we will try to convert arbitrary sources into ones of high min-entropy rate

Lemma 5.28. If X is an $(n - \Delta)$-source of length n, and $X = (X_1, X_2)$ is a partition of X into blocks of lengths n_1 and n_2, then for every $\varepsilon > 0$, (X_1, X_2) is ε-close to some $(n_1 - \Delta, n_2 - \Delta - \log(1/\varepsilon))$ block source.

Consider $\Delta = \alpha n$ for a constant $\alpha < 1/2$, and $n_1 = n_2 = n/2$. Then each block contributes min-entropy at least $(1/2 - \alpha)n$. The proofs of Lemmas 5.27 and 5.28 are left as exercises (Problem 5.1).

We are still left with the question of converting a general k-source into one of high min-entropy rate. We will do this via the following kind of object.

Definition 5.29. A function $\mathrm{Con} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a $k \to_\varepsilon k'$ *condenser* if for every k-source X on $\{0,1\}^n$, $\mathrm{Con}(X, U_d)$ is ε-close to some k'-source. Con is *lossless* if $k' = k + d$.

If $k'/m > k/n$, then the condenser increases the min-entropy rate, intuitively making extraction an easier task.

In Chapter 6, we will construct the following kind of condenser, using connections with both expander graphs and list-decodable error-correcting codes:

Theorem 5.30. For every constant $\alpha > 0$, for all positive integers $n \geq k$ and all $\varepsilon > 0$, there is an explicit

$$k \to_\varepsilon k + d$$

lossless condenser $C : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with $d = O(\log n + \log(1/\varepsilon))$ and $m = (1 + \alpha)k + O(\log(n/\varepsilon))$.

Note that setting α to be a small constant, we obtain an output min-entropy rate arbitrarily close to 1.

5.3.3 The Extractor

In this section, we will use the ideas outlined in the previous section — namely condensing and block-source extraction — to construct an extractor that is optimal up to constant factors (assuming the condenser of Theorem 5.30).

Theorem 5.31. Assume Theorem 5.30. Then for all positive integers $n \geq k$ and all $\varepsilon > 0$, there is an explicit (k, ε) extractor $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with $m \geq k/2$ and $d = O(\log(n/\varepsilon))$.

We will use the following building block, constructed in Problem 5.5.

Lemma 5.32. Assume Theorem 5.30. Then for every *constant* $t > 0$ and all positive integers $n \geq k$ and all $\varepsilon > 0$, there is an explicit (k, ε) extractor $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with $m \geq k/2$ and $d = k/t + O(\log(n/\varepsilon))$.

The point is that this extractor has a seed length that is an arbitrarily large constant factor (approximately $t/2$) smaller than its output length. Thus, if we use it as Ext_2 in the block-source extraction of Lemma 5.24, the resulting seed length will be smaller than that of Ext_1 by an arbitrarily large constant factor. (The seed length of the composed extractor Ext' in Lemma 5.24 is the same of that as Ext_2, which will be a constant factor smaller than its output length m_2, which we can take to be equal to the seed length d_1 of Ext_1.)

Overview of the Construction. Note that for small min-entropies k, namely $k = O(\log(n/\varepsilon))$, the extractor we want is already given by Lemma 5.32 with seed length d smaller than the output length m by any constant factor. (If we allow $d \geq m$, then extraction is trivial — just output the seed.) Thus, our goal will be to recursively construct extractors for large min-entropies using extractors for smaller min-entropies. Of course, if $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k_0, ε) extractor, say with $m = k_0/2$, then it is also a (k, ε) extractor for every $k \geq k_0$. The problem is that the output length is only $k_0/2$ rather than $k/2$. Thus, we need to increase the output length. This can be achieved by simply applying extractors for smaller min-entropies several times:

Lemma 5.33. Suppose $\mathrm{Ext}_1 : \{0,1\}^n \times \{0,1\}^{d_1} \to \{0,1\}^{m_1}$ is a (k_1, ε_1) extractor and $\mathrm{Ext}_2 : \{0,1\}^n \times \{0,1\}^{d_2} \to \{0,1\}^{m_2}$ is a (k_2, ε_2) extractor for $k_2 = k_1 - m_1 - \log(1/\varepsilon_3)$. Then $\mathrm{Ext}' : \{0,1\}^n \times \{0,1\}^{d_1+d_2} \to \{0,1\}^{m_1+m_2}$ is a $(k_1, \varepsilon_1 + \varepsilon_2 + \varepsilon_3)$ extractor.

The proof of this lemma follows from Lemma 5.27. After conditioning a k_1-source X on $W = \mathrm{Ext}_1(X, U_{d_1})$, X still has min-entropy at least $k_1 - m_1 - \log(1/\varepsilon_3) = k_2$ (except with probability ε_3), and thus $\mathrm{Ext}_2(X, U_{d_2})$ can extract an additional m_2 almost-uniform bits.

To see how we might apply this, consider setting $k_1 = .8k$ and $m_1 = k_1/2$, $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \varepsilon \geq 2^{-.1k}$, $k_2 = k_1 - m_1 - \log(1/\varepsilon_3) \in [.3k, .4k]$, and $m_2 = k_2/2$. Then we obtain a $(k, 3\varepsilon)$ extractor Ext' with output length $m = m_1 + m_2 > k/2$ from two extractors for min-entropies k_1, k_2 that are smaller than k by a constant factor, and we can hope to construct the latter two extractors recursively via the same construction.

Now, however, the problem is that the seed length grows by a constant factor in each level of recursion (e.g. if $d_1 = d_2 = d$ in Lemma 5.33, we get seed length $2d$ rather than d). Fortunately, as mentioned above, block source extraction using the extractor of Lemma 5.32 gives us a method to reduce the seed length by a constant factor. In order to apply block source extraction, we first need to convert our source to a block source; by Lemma 5.28, we can do this by using our condenser to make its entropy rate close to 1.

One remaining issue is that the error ε still grows by a constant factor in each level of recursion. However, we can start with polynomially small error at the base of the recursion and there are only logarithmically many levels of recursion, so we can afford this blow-up.

We now proceed with the proof details. It will be notationally convenient to do the steps in the reverse order from the description above — first we will reduce the seed length by a constant factor via block-source extraction, and then apply Lemma 5.33 to increase the output length.

Proof. [Theorem 5.31] Fix $n \in \mathbb{N}$ and $\varepsilon_0 > 0$. Set $d = c\log(n/\varepsilon_0)$ for an error parameter ε_0 and a sufficiently large constant c to be determined in the proof below. (To avoid ambiguity, we will keep the dependence on c explicit throughout the proof, and all big-Oh notation hides universal constants independent of c.) For $k \in [0, n]$, let $i(k)$ be the smallest nonnegative integer i such that $k \leq 2^i \cdot 8d$. This will be the level of recursion in which we handle min-entropy k; note that $i(k) \leq \log k \leq \log n$.

For every $k \in [0, n]$, we will construct an explicit $\mathrm{Ext}_k : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{k/2}$ that is a $(k, \varepsilon_{i(k)})$ extractor, for an appropriate sequence $\varepsilon_0 \leq \varepsilon_1 \leq \varepsilon_2 \cdots$. Note that we require the seed length to remain d and the fraction of min-entropy extracted to remain $1/2$ for all values of k. The construction will be by induction on $i(k)$.

Base Case: $i(k) = 0$, i.e. $k \leq 8d$. The construction of Ext_k follows from Lemma 5.32, setting $t = 9$ and taking c to be a sufficiently large constant.

Inductive Case: We construct Ext_k for $i(k) \geq 1$ from extractors $\mathrm{Ext}_{k'}$ with $i(k') < i(k)$ as follows. Given a k-source X of length n, Ext_k works as follows.

(1) We apply our condenser (Theorem 5.30) to convert X into a source X' that is ε_0-close to a k-source of length $(9/8)k + O(\log(n/\varepsilon_0))$. This requires a seed of length $O(\log(n/\varepsilon_0))$.

(2) We divide X' into two equal-sized halves (X_1, X_2). By Lemma 5.28, (X_1, X_2) is $2\varepsilon_0$-close to a $2 \times k'$ block source for

$$k' = k/2 - k/8 - O(\log(n/\varepsilon_0)) \ .$$

Note that $i(k') < i(k)$. Since $i(k) \geq 1$, we also have $k' \geq 3d - O(\log(n/\varepsilon_0)) \geq 2d$, for a sufficiently large choice of the constant c.

(3) Now we apply block-source extraction as in Lemma 5.24. We take Ext_2 to be a $(2d, \varepsilon_0)$ extractor from Lemma 5.32 with parameter $t = 16$, which will give us $m_2 = d$ output bits using a seed of length $d_2 = (2d)/16 + O(\log(n/\varepsilon_0))$. For Ext_1, we use our recursively constructed $\mathrm{Ext}_{k'}$, which has seed length d, error $\varepsilon_{i(k')}$, and output length $k'/2 \geq k/6$ (where the latter inequality holds for a sufficiently large choice of the constant c, because $k > 8d > 8c\log(1/\varepsilon)$).

All in all, our extractor so far has seed length at most $d/8 + O(\log(n/\varepsilon_0))$, error at most $\varepsilon_{i(k)-1} + O(\varepsilon_0)$, and output length at least $k/6$. This would be sufficient for our induction except that the output length is only $k/6$ rather than $k/2$. We remedy this by applying Lemma 5.33.

With one application of the extractor above, we extract at least $m_1 = k/6$ bits of the source min-entropy. Then with another application of the extractor above for min-entropy threshold $k_2 =$

$k - m_1 - \log(1/\varepsilon) = 5k/6 - \log(1/\varepsilon)$, by Lemma 5.33, we extract another $(5k/6 - \log(1/\varepsilon))/6$ bits and so on. After four applications, we have extracted all but $(5/6)^4 \cdot k + O(\log(1/\varepsilon)) \le k/2$ bits of the min-entropy. Our seed length is then $4 \cdot (d/8 + O(\log(n/\varepsilon_0))) \le d$ and the total error is $\varepsilon_{i(k)} = O(\varepsilon_{i(k)-1})$.

Solving the recurrence for the error, we get $\varepsilon_i = 2^{O(i)} \cdot \varepsilon_0 \le \text{poly}(n) \cdot \varepsilon_0$, so we can obtain error ε by setting $\varepsilon_0 = \varepsilon/\text{poly}(n)$. As far as explicitness, we note that computing Ext_k consists of four evaluations of our condenser from Theorem 5.30, four evaluations of $\text{Ext}_{k'}$ for values of k' such that $i(k') < (i(k) - 1)$, four evaluations of the explicit extractor from Lemma 5.32, and simple string manipulations that can be done in time $\text{poly}(n, d)$. Thus, the total computation time is at most $4^{i(k)} \cdot \text{poly}(n, d) = \text{poly}(n, d)$. □

Repeatedly applying Lemma 5.33 using extractors from Theorem 5.31, we can extract any constant fraction of the min-entropy using a logarithmic seed length, and all the min-entropy using a polylogarithmic seed length.

Corollary 5.34. Assume Theorem 5.30. Then the following holds for every constant $\alpha > 0$. For every $n \in \mathbb{N}$, $k \in [0, n]$, and $\varepsilon > 0$, there is an explicit (k, ε) extractor $\text{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with $m \ge (1 - \alpha)k$ and $d = O(\log(n/\varepsilon))$.

Corollary 5.35. Assume Theorem 5.30. Then for every $n \in \mathbb{N}$, $k \in [0, n]$, and $\varepsilon > 0$, there is an explicit (k, ε) extractor $\text{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with $m = k - O(\log(1/\varepsilon))$ and $d = O(\log k \cdot \log(n/\varepsilon))$.

We remark that the above construction can be modified to yield *strong* extractors achieving the same output lengths as above (so the entropy of the seed need not be lost in Corollary 5.35).

A summary of the extractor parameters we have seen is in Table 5.1.

Method	Seed Length d	Output Length m
Optimal and Nonconstructive	$\log(n - k) + O(1)$	$k + d - O(1)$
Necessary for **BPP** Simulation	$O(\log n)$	$k^{\Omega(1)}$
Spectral Expanders	$O(n - k)$	n
Pairwise Independent Hashing	$O(n)$	$k + d - O(1)$
Corollary 5.34	$O(\log n)$	$(1 - \gamma)k$, any constant $\gamma > 0$
Corollary 5.35	$O(\log^2 n)$	$k - O(1)$

Table 5.1 Parameters for some constructions of $(k, .01)$ extractors.

While Theorem 5.31 and Corollary 5.34 give extractors that are optimal up to constant factors in both the seed length and output length, it remains an important open problem to get one or both of these to be optimal to within an *additive* constants while keeping the other optimal to within a constant factor.

Open Problem 5.36. Give an explicit construction of $(k, .01)$ extractors $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with seed length $d = O(\log n)$ and output length $m = k+d-O(1)$ (or even $m = (1-o(1))\cdot k$).

By using the condenser of Theorem 5.30, it suffices to solve achieve the above for high min-entropy rate, e.g. $k = .99n$.

Open Problem 5.37. Give an explicit construction of $(k, .01)$ extractors $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with seed length $d = \log n + O(1)$ and $m = \Omega(k)$ (or even $m = k^{\Omega(1)}$).

One of the reasons that these open problems are significant is that, in many applications of extractors, the resulting complexity depends *exponentially* on the seed length d and/or the entropy loss $k + d - m$. (An example is the simulation of **BPP** with weak random sources given by Proposition 5.15.) Thus, additive constants in these parameters corresponds to constant factors in complexity.

Another open problem is more aesthetic in nature. The construction of Theorem 5.31 makes use of the condenser of Theorem 5.31, the Leftover Hash Lemma (Theorem 5.18) and the composition techniques of Lemmas 5.24 and Lemma 5.33 in a somewhat complex recursion. It is of interest to have a construction that is more direct. In addition to the aesthetic appeal, such a construction would likely be more practical to implement and provide more insight into extractors. In Chapter 7, we will see a very direct construction based on a connection between extractors and pseudorandom generators, but its parameters will be somewhat worse than Theorem 5.31. Thus the following remains open:

Theorem 5.38. Give a "direct" construction of (k, ε) extractors $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with seed length $d = O(\log(n/\varepsilon))$ and $m = \Omega(k)$.

5.4 More Connections with Expanders

5.4.1 Lossless Condensers vs. Expanders

In the previous section, we saw the notion of a $k \to_\varepsilon k'$ *condenser* $\mathrm{Con} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, where for every k-source X on $\{0,1\}^n$, $\mathrm{Con}(X, U_d)$ is ε-close to some k'-source. We can define the *entropy loss* of a condenser to be $\ell = k + d - k'$. As we have discussed, an extractor (i.e. $m = k'$) must have $\ell \geq 2\log(1/\varepsilon) - O(1)$, whereas if we allow m to be larger than k' (specifically, $m \geq k' + \log(1/\varepsilon) + O(1)$), then it is possible for a condenser to be *lossless* (i.e. have $\ell = 0$).

As we have seen for extractors in Section 5.2.2, every function $\mathrm{Con} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ can be viewed as a bipartite multigraph G with $N = 2^n$ left vertices, left-degree $D = 2^d$, and $M = 2^m$ right-vertices where the y'th neighbor of left-vertex x is $\mathrm{Con}(x, y)$. Generalizing what we showed for extractors, the condenser property implies a vertex-expansion property of the graph G. Specifically, recalling that an $(=K, A)$ vertex expander is one in which sets of size exactly K expand by a factor of A, then we have:

Proposition 5.39. If $\mathrm{Con} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a $k \to_\varepsilon k'$ condenser, then the corresponding bipartite graph is a $(= K, A)$ vertex expander for $A = (1-\varepsilon) \cdot K'/K = (1-\varepsilon) \cdot D/L$, where $K = 2^k$, $K' = 2^{k'}$, $D = 2^d$, $L = 2^\ell$, and $\ell = k + d - k'$ is the entropy loss of Con.

Thus, if $L < (1-\varepsilon) \cdot D$, the expansion factor A is in fact bigger than 1 (but the case $A < 1$ is still interesting and nontrivial, because the graph is unbalanced). In general, the vertex expansion property is weaker than the property of being a condenser — for example, when $m = k'$, it corresponds to a *disperser* rather than an extractor (Proposition 5.20). However, for the special case of *lossless* condensers (i.e. $L = 1$), it turns out that the two properties are equivalent.

Proposition 5.40. A function $\mathrm{Con} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a $k \to_\varepsilon k+d$ (lossless) condenser *if and only if* the corresponding bipartite graph is a $(= K, A)$ vertex expander for $A = (1-\varepsilon) \cdot D$, where $K = 2^k$, $K' = 2^{k'}$, and $D = 2^d$ (provided $2^k \in \mathbb{N}$).

Proof. The "only if" direction follows from Proposition 5.39, so we only prove the "if" direction. Assume that the bipartite graph corresponding to Con is a $(= K, (1-\varepsilon) \cdot D)$ vertex expander. To show that Con is a lossless condenser, by Lemma 5.10 it suffices to show that $\mathrm{Con}(X, U_d)$ is ε-close to a k'-source for every *flat* k-source X. Let S be the support of X, which is of size K. By the vertex expansion of the graph, $|N(S)| \geq (1-\varepsilon) \cdot DK$. Since there are only DK edges leaving S, we can make all of these edges lead to distinct vertices by shifting an ε fraction of them. Let $T \subset [M]$ be the set of KD vertices hit after this shifting. Then $\mathrm{Con}(X, U_d)$ is ε-close to the uniform distribution on T, which is a $(k + d)$-source. □

Thus, the lossless condenser that we assumed in the previous lecture follows immediately from the following expander:

Theorem 5.41. For every constant $\alpha > 0$, every $N \in \mathbb{N}$, $K \leq N$, and $\varepsilon > 0$, there is an explicit $(K, (1-\varepsilon)D)$ expander with N left-vertices, M right-vertices, left-degree $D = O((\log N)(\log K)/\varepsilon)^{1+1/\alpha}$ and $M \leq D^2 \cdot K^{1+\alpha}$. Moreover, D is a power of 2.

We will construct this expander in Chapter 6, using ideas based on list-decodable error-correcting codes.

Note that the kind of expander given by this theorem can be used for the data structure application in Problem 4.6 — storing a $K/2$-sized subset $S \subseteq [N]$ with $M = K^{1+\alpha} \cdot \mathrm{polylog}(N)$ bits in such a way that membership can be probabilistically tested by reading only 1 bit of the data structure. (An efficient solution to this application actually requires more than the graph being explicit in the usual sense, but also that there are efficient algorithms for finding all left-vertices having at least a δ fraction neighbors in a given set $T \subseteq [M]$ of right vertices, but it turns out that the expanders we will construct have this property.)

A deficiency of the expander of Theorem 5.41 is that the size of the right-hand side is polynomial in K and D (for constant α), whereas the optimal bound is $M = O(KD/\varepsilon)$. Achieving the latter, while keeping the left-degree polylogarithmic, is an open problem:

Open Problem 5.42. Construct $(= K, A)$ bipartite expanders with N left-vertices, degree $D = \mathrm{poly}(\log N)$, expansion $A = .99D$, and $M = O(KD)$ right-hand vertices.

We remark that a construction where D is *quasipolynomial* in $\log N$ is known. By Proposition 5.40, a solution to Open Problem 5.42 would give lossless condensers whose output is of extremely high min-entropy ($k' = m - O(1)$), and thus we could get extractors that extract all the min-entropy by then applying extractors based on spectral expanders (Theorem 5.22), thereby also solving Open Problem 5.36.

5.4.2 Block-Source Extraction vs. the Zig-Zag Product

Recall the block-source extraction method presented last time. We define $\mathrm{Ext}' : \{0,1\}^{n_1+n_2} \times \{0,1\}^{d_2} \to \{0,1\}^{m_1}$ by $\mathrm{Ext}'((x_1,x_2),y_2) = \mathrm{Ext}_1(x_1, \mathrm{Ext}_2(x_2,y_2))$. (Here we consider the special case that $m_2 = d_1$.)

Viewing the extractors as bipartite graphs, the left-vertex set is $[N_1] \times [N_2]$ and the left-degree is D_2. A random step from a vertex $(x_1,x_2) \in [N_1] \times [N_2]$ corresponds to taking a random step from x_2 in G_2 to obtain a right-hand vertex $y_1 \in \{0,1\}^{m_2}$, which we view as an edge label y for G_1. We then move to the y'th neighbor of x_1.

This is just like the first two steps of the zig-zag graph product. Why do we need a third step in the zig-zag product? It is because of the slightly different goals in the two setting. In a (spectral) expander, we consider an arbitrary initial distribution that does not have too much (Renyi) entropy, and need to add entropy to it. In a block-source extractor, our initial distribution is constrained to be a block source (so both blocks have a certain amount of min-entropy), and our goal is to produce an almost-uniform output (even if we end up with less bits than the initial entropy).

Thus, in the zig-zag setting, we must consider the following extreme cases (that are ruled out for block sources):

- The second block has no entropy given the first. Here, the step using G_2 will add entropy, but not enough to make y_1 close to uniform. Thus, we have no guarantees on the behavior of the G_1-step, and we may lose entropy with it. For this reason, we keep track of the edge used in the G_1-step — that is, we remember b_1 such that x_1 is the b_1'th neighbor of $z_1 = \mathrm{Ext}(x_1,y_1)$. This ensures that the (edge-rotation) mapping $(x_1,y_1) \mapsto (z_1,b_1)$ is a permutation and does not lose any entropy. We can think of b_1 as a "buffer" that retains any extra entropy in (x_1,y_1) that did not get extracted into z_1. So a natural idea is to just do block source extraction, but output (z_1,b_1) rather than just z_1. However, this runs into trouble with the next case.
- The first block has no entropy but the second block is completely uniform given the first. In this case, the G_2 step cannot add any entropy and the G_1 step does not add any entropy because it is a permutation. However, the G_1 step transfers entropy into z_1. So if we add another expander-step from b_1 at the end, we can argue that it will add entropy. This gives rise to the 3-step definition of the zig-zag product.

While we analyzed the zig-zag product with respect to spectral expansion (i.e. Rényi entropy), it is also possible to do analyze it in terms of a condenser-like definition (i.e. distributions ε-close to having some min-entropy). all" It turns out that a variant of the zig-zag product for condensers leads to a construction of constant-degree bipartite expanders with expansion $(1-\varepsilon)\cdot D$ for the balanced ($M=N$) or slightly unbalanced (e.g. $M=\Omega(N)$) case. However, as mentioned in Open Problems 4.39, 4.40, and 5.42, there are still several significant open problems concerning the explicit construction of expanders with vertex expansion close to the degree, involving achieving expansion $D-O(1)$, the non-bipartite case, and achieving a near-optimal number of right-hand vertices.

5.5 Exercises

~~**Problem 5.1.** (Min-entropy and Statistical Difference)~~

(1) Prove that for every two random variables X and Y,

$$\Delta(X,Y)=\max_f|\operatorname{E}[f(X)]-\operatorname{E}[f(Y)]|=\frac{1}{2}\cdot|X-Y|_1,$$

where the maximum is over all $[0,1]$-valued functions f. (Hint: first identify the functions f that maximize $|\operatorname{E}[f(X)]-\operatorname{E}[f(Y)]|$.)

(2) Suppose that (W,X) are jointly distributed random variables where W takes values in $\{0,1\}^\ell$ and (W,X) is a k-source. Show that for every $\varepsilon>0$, with probability at least $1-\varepsilon$ over $w\overset{\text{R}}{\leftarrow}W$, we have $X|_{W=w}$ is a $(k-\ell-\log(1/\varepsilon))$-source.

(3) Suppose that X is an $(n-\Delta)$-source taking values in $\{0,1\}^n$, and we let X_1 consist of the first n_1 bits of X and X_2 the remaining $n_2=n-n_1$ bits. Show that for every $\varepsilon>0$, (X_1,X_2) is ε-close to some $(n_1-\Delta,n_2-\Delta-\log(1/\varepsilon))$ block source.

Problem 5.2. (Extractors vs. Samplers) One of the problems we have revisited several times is that of randomness-efficient sampling: Given oracle access to a function $f:\{0,1\}^m\to[0,1]$, approximate its average value $\mu(f)$ to within some small additive error. Most of the samplers we have seen work as follows: they choose some n random bits, use these to decide on some D samples $z_1,\ldots,z_D\in\{0,1\}^m$, and output the average of $f(z_1),\ldots,f(z_D)$. We call such a procedure a (δ,ε)-*averaging sampler* if, for any function f, the probability that the sampler's output differs from $\mu(f)$ by more than ε is at most δ. In this problem, we will see that averaging samplers are essentially equivalent to extractors.

Given $\mathrm{Ext}\colon\{0,1\}^n\times\{0,1\}^d\to\{0,1\}^m$, we obtain a sampler Smp which chooses $x\overset{\text{R}}{\leftarrow}\{0,1\}^n$, and uses $\{\mathrm{Ext}(x,y):y\in\{0,1\}^d\}$ as its $D=2^d$ samples. Conversely, every sampler Smp using n random bits to produce $D=2^d$ samples in $\{0,1\}^m$ defines a function $\mathrm{Ext}\colon\{0,1\}^n\times\{0,1\}^d\to\{0,1\}^m$.

(1) Prove that if Ext is a $(k-1,\varepsilon)$-extractor, then Smp is a $(2^k/2^n,\varepsilon)$-averaging sampler.

(2) Prove that if Smp is a $(2^k/2^n,\varepsilon)$-sampler, then Ext is a $(k+\log(1/\varepsilon),2\varepsilon)$-extractor.

(3) Suppose we are given a constant-error **BPP** algorithm which uses $r=r(n)$ random bits on inputs of length n. Show how, using Part 1 and the extractor of Theorem 5.31, we can reduce its error probability to $2^{-\ell}$ using $O(r)+\ell$ random bits, for any polynomial $\ell=\ell(n)$. (Note that this improves the $r+O(\ell)$ given by expander walks for $\ell\gg r$.) Conclude that every problem in **BPP** has a randomized algorithm which only errs for $2^{q^{0.01}}$ choices of its q random bits!

Problem 5.3. (Encryption and Deterministic Extraction) A (one-time) *encryption scheme* with key length n and message length m consists of an encryption function $\mathrm{Enc}\colon \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^\ell$ and a decryption function $\mathrm{Dec}\colon \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^m$ such that $\mathrm{Dec}(k, \mathrm{Enc}(k,u)) = u$ for every $k \in \{0,1\}^n$ and $u \in \{0,1\}^m$. Let K be a random variable taking values in $\{0,1\}^n$. We say that $(\mathrm{Enc}, \mathrm{Dec})$ is *(statistically) ε-secure with respect to K* if for every two messages $u, v \in \{0,1\}^m$, we have $\Delta(\mathrm{Enc}(K,u), \mathrm{Enc}(K,v)) \le \varepsilon$. For example, the *one-time pad*, where $n = m = \ell$ and $\mathrm{Enc}(k,u) = k \oplus u = \mathrm{Dec}(k,u)$ is 0-secure (aka perfectly secure) with respect to the uniform distribution $K = U_m$. For a class $\mathcal{C}$ of sources on $\{0,1\}^n$, we say that the encryption scheme $(\mathrm{Enc}, \mathrm{Dec})$ is *ε-secure with respect to $\mathcal{C}$* if Enc is ε-secure with respect to every $K \in \mathcal{C}$.

(1) Show that if there exists a deterministic ε-extractor $\mathrm{Ext}\colon \{0,1\}^n \to \{0,1\}^m$ for $\mathcal{C}$, then there exists an 2ε-secure encryption scheme with respect to $\mathcal{C}$.

(2) Conversely, use the following steps to show that if there exists an ε-secure encryption scheme $(\mathrm{Enc}, \mathrm{Dec})$ with respect to $\mathcal{C}$, where $\mathrm{Enc}\colon \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^\ell$, then there exists a deterministic 2ε-extractor $\mathrm{Ext}\colon \{0,1\}^n \to \{0,1\}^{m - 2\log(1/\varepsilon) - O(1)}$ for $\mathcal{C}$, provided $m \ge \log n + 2\log(1/\varepsilon) + O(1)$.

   (a) For each fixed key $k \in \{0,1\}^n$, define a source X_k on $\{0,1\}^\ell$ by $X_k = \mathrm{Enc}(k, U_m)$, and let $\mathcal{C}'$ be the class of all these sources (i.e., $\mathcal{C}' = \{X_k : k \in \{0,1\}^n\}$). Show that there exists a deterministic ε-extractor $\mathrm{Ext}'\colon \{0,1\}^\ell \to \{0,1\}^{m - 2\log(1/\varepsilon) - O(1)}$ for $\mathcal{C}'$, provided $m \ge \log n + 2\log(1/\varepsilon) + O(1)$.

   (b) Show that if Ext' is a deterministic ε-extractor for $\mathcal{C}'$ and Enc is ε-secure with respect to $\mathcal{C}$, then $\mathrm{Ext}(k) = \mathrm{Ext}'(\mathrm{Enc}(k, 0^m))$ is a deterministic 2ε-extractor for $\mathcal{C}$.

Thus, a class of sources can be used for secure encryption iff it is deterministically extractable.

Problem 5.4. (Extracting from Unpredictable-Bit Sources)

(1) Let X be a source taking values in $\{0,1\}^n$ such that for all x, y, $\Pr[X = x]/\Pr[X = y] \le (1-\delta)/\delta$. Show that $X \in \mathrm{UnpredBits}_{n,\delta}$.

(2) Prove that for every function $\mathrm{Ext}\colon \{0,1\}^n \to \{0,1\}$ and every $\delta > 0$, there exists a source $X \in \mathrm{UnpredBits}_{n,\delta}$ with parameter δ such that $\Pr[\mathrm{Ext}(X) = 1] \le \delta$ or $\Pr[\mathrm{Ext}(X) = 0] \ge 1 - \delta$. (Hint: for $b \in \{0,1\}$, consider X that is uniform on $\mathrm{Ext}^{-1}(b)$ with probability $1-\delta$ and is uniform on $\mathrm{Ext}^{-1}(\bar{b})$ with probability δ.)

(3) (*) Show how to extract from sources in $\mathrm{UnpredBits}_{n,\delta}$ using a seeded extractor with a seed of *constant* length. That is, the seed length should not depend on the length n of the source, but only on the bias parameter δ and the statistical difference ε from uniform desired. The number of bits extracted should be $\Omega(\delta n)$.

Problem 5.5. (The Building-Block Extractor) Assume the condenser stated in Theorem 5.30. Show that for every *constant* $t > 0$ and all positive integers $n \geq k$ and all $\varepsilon > 0$, there is an *explicit* (k, ε)-extractor $\mathrm{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with $m = k/2$ and $d = k/t + O(\log(n/\varepsilon))$. (Hint: convert the source into a block source with blocks of length $k/O(t) + O(\log(n/\varepsilon))$.)

Problem 5.6. (Extracting from Symbol-Fixing Sources*) A generalization of a bit-fixing source is a *symbol-fixing source* X taking values in Σ^n for some alphabet Σ, where subset of the coordinates of X are fixed and the rest are uniformly distributed and independent elements of Σ. For $\Sigma = \{0, 1, 2\}$ and $k \in [0, n]$, give an explicit ε-extractor $\mathrm{Ext} : \Sigma^n \to \{0,1\}^m$ for the class of of symbol-fixing sources on Σ^n with min-entropy at least k, with $m = \Omega(k)$ and $\varepsilon = 2^{-\Omega(k)}$. (Hint: use a random walk on a consistently labelled 3-regular expander graph.)