

## Chapter 2

# Deterministic Extractors for Bit-Fixing Sources by Obtaining an Independent Seed

### Summary

An  $(n, k)$ -bit-fixing source is a distribution  $X$  over  $\{0, 1\}^n$  such that there is a subset of  $k$  variables in  $X_1, \dots, X_n$  which are uniformly distributed and independent of each other, and the remaining  $n - k$  variables are fixed. A deterministic bit-fixing source extractor is a function  $E : \{0, 1\}^n \rightarrow \{0, 1\}^m$  which on an arbitrary  $(n, k)$ -bit-fixing source outputs  $m$  bits that are statistically-close to uniform. Prior to our work, Kamp and Zuckerman [44th FOCS, 2003] gave a construction of a deterministic bit-fixing source extractor that extracts  $\Omega(k^2/n)$  bits and requires  $k > \sqrt{n}$ .

In this chapter we give constructions of deterministic bit-fixing source extractors that extract  $(1 - o(1))k$  bits whenever  $k > (\log n)^c$  for some universal constant  $c > 0$ . Thus, our constructions extract almost all the randomness from bit-fixing sources and work even when  $k$  is small. For  $k \gg \sqrt{n}$  the extracted bits have statistical distance  $2^{-n^{\Omega(1)}}$  from uniform, and for  $k \leq \sqrt{n}$  the extracted bits have statistical distance  $k^{-\Omega(1)}$  from uniform.

Our technique gives a general method to transform deterministic bit-fixing source extractors that extract few bits into extractors which extract almost all the bits. This work is the first to use the ‘recycling paradigm’ as described in the introduction. The description of it here is different and perhaps more cumbersome, as the one given in the introduction was only realized in hindsight.

This chapter is based on [26].

## 2.1 Introduction

### 2.1.1 Bit-Fixing Sources

In this chapter we concentrate on the family of “bit-fixing sources” introduced by Chor et al. [15]. A distribution  $X$  over  $\{0, 1\}^n$  is a bit-fixing source if there is a subset  $S \subseteq \{1, \dots, n\}$  of “good indices” such that the bits  $X_i$  for  $i \in S$  are independent fair coins and the rest of the bits are fixed.<sup>1</sup>

**Definition 2.1** (bit-fixing sources and extractors). *A distribution  $X$  over  $\{0, 1\}^n$  is an  $(n, k)$ -bit-fixing source if there exists a subset  $S = \{i_1, \dots, i_k\} \subseteq \{1, \dots, n\}$  such that  $X_{i_1}, X_{i_2}, \dots, X_{i_k}$  is uniformly distributed over  $\{0, 1\}^k$  and for every  $i \notin S$ ,  $X_i$  is constant.*

*A function  $E : \{0, 1\}^n \rightarrow \{0, 1\}^m$  is a deterministic  $(k, \epsilon)$ -bit-fixing source extractor if it is a deterministic  $\epsilon$ -extractor (as defined in the first chapter) for all  $(n, k)$ -bit-fixing sources.*

One of the motivations given in the literature for studying deterministic bit-fixing source extractors is that they are helpful in cryptographic scenarios in which an adversary learns (or alters)  $n - k$  bits of an  $n$ -bit long secret key [15]. Loosely speaking, one wants cryptographic protocols to remain secure even in the presence of such adversaries. Various models for such “exposure resilient cryptography” were studied [57, 11, 12, 20]. The reader is referred to [18] for a comprehensive treatment of “exposure resilient cryptography” and its relation to deterministic bit-fixing source extractors.

Every  $(n, k)$ -bit-fixing source “contains”  $k$  “bits of randomness”. It follows that any deterministic  $(k, \epsilon)$ -bit-fixing source extractor with  $\epsilon < 1/2$  can extract at most  $k$  bits. The function  $E(x) = \bigoplus_{1 \leq i \leq n} x_i$  is a deterministic  $(k, 0)$ -bit-fixing source extractor which extracts one bit for any  $k \geq 1$ . Chor et al. [15] concentrated on deterministic “errorless” extractors (that is, deterministic extractors in which  $\epsilon = 0$ ). They show that such extractors cannot extract even two bits when  $k < n/3$ . They also give some constructions of deterministic errorless extractors for large  $k$ .

Our focus is on extractors with error  $\epsilon > 0$  (which allows extracting many bits for many choices of  $k$ ). A probabilistic argument shows the existence of a deterministic  $(k, \epsilon)$ -bit-fixing source extractor that extracts  $m = k - O(\log(n/\epsilon))$  bits for any choice of  $k$  and  $\epsilon$ . Thus, it is natural to try and achieve such parameters by explicit constructions.

In a previous work, Kamp and Zuckerman [37] constructed explicit deterministic  $(k, \epsilon)$ -bit-fixing source extractors that extract  $m = \eta k^2/n$  bits for some constant  $0 < \eta < 1$  with  $\epsilon = 2^{-\Omega(k^2/n)}$ . They pose the open problem to

---

<sup>1</sup>We remark that such sources are often referred to as “oblivious bit-fixing sources” to differentiate them from other types of “non-oblivious” bit-fixing sources in which the bits outside of  $S$  may depend on the bits in  $S$  (cf. [7]). In this chapter we are only concerned with the “oblivious case”.

extract more bits from such sources. Note that the extractor of Kamp and Zuckerman is inferior to the nonexplicit extractor in two respects:

- It only works when  $k > \sqrt{n}$ .
- Even when  $k > \sqrt{n}$  the extractor may extract only a small fraction of the randomness. For example, if  $k = n^{1/2+\alpha}$  for some  $0 < \alpha < 1/2$  the extractor only extracts  $m = \eta n^{2\alpha}$  bits.

### 2.1.2 Our Results

We give two constructions of deterministic bit-fixing source extractors that extract  $m = (1 - o(1))k$  bits from  $(n, k)$ -bit-fixing sources. Our first construction is for the case of  $k \gg \sqrt{n}$ .

**Theorem 2.1.** *For every constant  $0 < \gamma < 1/2$  there exists an integer  $n'$  (depending on  $\gamma$ ) such that for any  $n > n'$  and any  $k$ , there is an explicit deterministic  $(k, \epsilon)$ -bit-fixing source extractor  $E : \{0, 1\}^n \rightarrow \{0, 1\}^m$  where  $m = k - n^{1/2+\gamma}$  and  $\epsilon = 2^{-\Omega(n^\gamma)}$ .*

Consider  $k = n^{1/2+\alpha}$  for some constant  $0 < \alpha < 1/2$ . We can choose any  $\gamma < \alpha$  and extract  $m = n^{1/2+\alpha} - n^{1/2+\gamma}$  bits whereas the construction of [37] only extracts  $m = O(n^{2\alpha})$  bits. For this choice of parameters we achieve error  $\epsilon = 2^{-\Omega(n^\gamma)}$  whereas [37] achieves a slightly smaller error  $\epsilon = 2^{-\Omega(n^{2\alpha})}$ . We remark that this comes close to the parameters achieved by the nonexplicit construction, which can extract  $m = n^{1/2+\alpha} - n^{1/2+\gamma}$  with error  $\epsilon = 2^{-\Omega(n^{1/2+\gamma})}$ .

Our second construction works for any  $k > (\log n)^c$  for some universal constant  $c$ . However, the error in this construction is larger.

**Theorem 2.2.** *There exist constants  $c > 0$  and  $0 < \mu, \nu < 1$  such that for any large enough  $n$  and any  $k \geq \log^c n$ , there is an explicit deterministic  $(k, \epsilon)$ -bit-fixing source extractor  $E : \{0, 1\}^n \rightarrow \{0, 1\}^m$  where  $m = k - O(k^\nu)$  and  $\epsilon = O(k^{-\mu})$ .*

We remark that using the technique of [37] one can achieve much smaller error ( $\epsilon = 2^{-\sqrt{k}}$ ) at the cost of extracting very few bits ( $m = \Omega(\log k)$ ). The precise details are given in Theorem 2.5.

### 2.1.3 Overview of Techniques

We develop a general technique that transforms any deterministic bit-fixing source extractor that extracts only very few bits into one that extracts almost all of the randomness in the source. This transformation makes use of “seeded extractors”.

### Seeded randomness extractors

A seeded randomness extractor is a function which receives two inputs: In addition to a sample from a source  $X$ , a seeded extractor also receives a short “seed”  $Y$  of few uniformly distributed bits. Loosely speaking, the extractor is required to output many more random bits than the number of bits “invested” as a seed.

**Definition 2.2** (seeded extractors). *Let  $\mathcal{C}$  be a class of distributions on  $\{0, 1\}^n$ . A function  $E : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  is a seeded  $\epsilon$ -extractor for  $\mathcal{C}$  if for every source  $X$  in  $\mathcal{C}$  the distribution  $E(X, Y)$  (obtained by sampling  $x$  from  $X$  and a uniform  $y \in \{0, 1\}^d$  and computing  $E(x, y)$ ) is  $\epsilon$ -close to the uniform distribution on  $m$  bit strings.*

A long line of research focuses on constructing such seeded extractors with as short as possible seed length that extract as many as possible bits from the most general family of sources that allow randomness extraction: The class of sources with high min-entropy.

**Definition 2.3** (seeded extractors for high min-entropy sources). *The min-entropy of a distribution  $X$  over  $\{0, 1\}^n$  is  $H_\infty(X) = \min_{x \in \{0, 1\}^n} \log_2(1/\Pr(x))$ . A function  $E : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  is a  $(k, \epsilon)$ -extractor if it is a seeded  $\epsilon$ -extractor for the class of all sources  $X$  with  $H_\infty(X) \geq k$ .*

There are explicit constructions of  $(k, \epsilon)$ -extractors that use a seed of length  $\text{polylog}(n/\epsilon)$  to extract  $k$  random bits. The reader is referred to [62] for a detailed survey on various constructions of seeded extractors.

Our goal is to construct *deterministic* bit-fixing source extractors. Nevertheless, in the next definition we introduce the concept of a *seeded* bit-fixing source extractor. We use such extractors as a component in our construction of deterministic bit-fixing source extractors.

**Definition 2.4** (seeded extractors for bit-fixing sources). *A function  $E : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  is a seeded  $(k, \epsilon)$ -bit-fixing source extractor if it is a seeded  $\epsilon$ -extractor for the class of all  $(n, k)$ -bit-fixing sources.*

### Seed obtainers

There is a very natural way to try to transform a deterministic bit-fixing source extractor that extracts few (say  $\text{polylog}(n)$ ) bits into one that extracts many bits: First, run the deterministic bit-fixing source extractor to extract a few bits from the source, and then use these bits as a seed to a seeded extractor that extracts all the bits from the source. The obvious difficulty with this approach is that typically the output of the first extractor is *correlated* with the imperfect random source. Seeded extractors are only guaranteed to work when their seed is *independent* from the random source. To overcome this difficulty we introduce a new object we call a “seed obtainer”.

Loosely speaking, a seed obtainer is a function  $F$  that, given an  $(n, k)$ -bit-fixing source  $X$ , outputs two strings  $X'$  and  $Y$  with the following properties:

- $X'$  is an  $(n, k')$ -bit-fixing source with  $k' \approx k$  good bits.
- $Y$  is a short string that is almost uniformly distributed.
- $X'$  and  $Y$  are almost independent.

The precise definition is slightly more technical and is given in Definition 2.11. Note that a seed obtainer reduces the task of constructing *deterministic* extractors to that of constructing *seeded* extractors: Given a bit-fixing source  $X$ , one first runs the seed obtainer to obtain  $X'$  and a short  $Y$ , and then uses  $Y$  as a seed to a seeded extractor that extracts all the randomness from  $X'$ . (In fact, it is even sufficient to construct seeded extractors for bit-fixing sources.)

### Constructing seed obtainers

Note that every seed obtainer  $F(X) = (X', Y)$  “contains” a deterministic bit-fixing source extractor by setting  $E(X) = Y$ . We show how to transform any deterministic bit-fixing source extractor into a seed obtainer. In this transformation the length of the “generated seed”  $Y$  is roughly the length of the output of the original extractor.

It is helpful to explain the intuition behind this transformation when applied to a specific deterministic bit-fixing source extractor. Consider the “xor-extractor”  $E(x) = \oplus_{1 \leq i \leq n} x_i$ . Let  $X$  be some  $(n, k)$ -bit-fixing source, and let  $Z = E(X)$ . Note that the output bit  $Z$  is indeed very correlated with the input  $X$ . Nevertheless, suppose that we somehow obtain a random small subset of the indices of  $X$ . It is expected that the set contains a small fraction of the good bits. Let  $X'$  be the string that remains after “removing” the indices in the sampled set. The important observation is that  $X'$  is a bit-fixing source that is *independent* from the output  $Z$ . It turns out that the same phenomenon happens for every deterministic bit-fixing source extractor  $E(X)$ . However, it is not clear how to use this idea as we don’t have additional random bits to perform the aforementioned sampling of a random set. Surprisingly, we show how to use the bits extracted by the extractor  $E$  to perform this sampling.

Following this intuition, given an extractor  $E(X)$  which extracts an  $m$  bit string  $Z$ , we partition  $Z$  into two parts  $Y$  and  $W$ . We then use  $W$  as a seed to a randomness efficient method of “sampling” a small subset  $T$  of  $\{1, \dots, n\}$ . The first output of the seed obtainer  $X'$  is given by “removing” the sampled indices from  $X$ . More formally,  $X'$  is the string  $X$  restricted to the indices outside of  $T$ . The second output is  $Y$  (the other part of the output of the extractor  $E$ ).

The intuition is that if  $T$  was a size  $n/r$  uniformly distributed subset of  $\{1, \dots, n\}$  then it is expected to hit approximately  $k/r$  good bits from the

source. Thus,  $k - k/r$  good bits remain in  $X'$ . We will require that the extractor  $E$  extracts randomness from  $(n, k/r)$ -bit-fixing sources. Loosely speaking, we can hope that  $E$  will extract its output from  $X_T$  (the string obtained by restricting  $X$  to the indices of  $T$ ). Thus, its output will be independent from  $X'$  (the string obtained by removing  $X_T$ ).

Note that the intuition above is far from being precise. The set  $T$  is sampled using random bits  $W$  that are extracted from the source  $X$ , and thus  $T$  depends on  $X$ , whereas, the intuition corresponds to the case where  $T$  is independent from  $X$ . The precise argument appears in Section 2.3. We remark that the analysis requires that the extractor  $E$  have error  $\epsilon$  that is smaller than  $2^{-|W|}$  (where  $|W|$  is the number of bits used by the sampling method).

### A deterministic extractor for large $k$ (i.e., $k \gg \sqrt{n}$ )

Our first construction builds on the deterministic bit-fixing source extractor of Kamp and Zuckerman [37] that works for  $k > \sqrt{n}$  and extracts at least  $\Omega(k^2/n)$  bits from the source. We first transform this extractor into a seed obtainer  $F$ . Next, we run the seed obtainer  $F$  on the input source to generate a bit-fixing source  $X'$  and a seed  $Y$ . Finally, we extract all the randomness in  $X'$  by running a seeded extractor on  $X'$  using  $Y$  as seed.

### A deterministic extractor for small $k$ (i.e., $k < \sqrt{n}$ )

In order to use our technique for  $k < \sqrt{n}$  we need to start with some deterministic bit-fixing source extractor that works when  $k < \sqrt{n}$  and extracts a small number of bits. Our first observation is that methods similar to the ones of Kamp and Zuckerman [37] can be applied when  $k < \sqrt{n}$  but only give deterministic bit-fixing source extractors that extract very few bits (i.e.,  $\Omega(\log k)$  bits)<sup>2</sup>.

**Deterministic extractors that extract  $\Omega(\log k)$  bits** Kamp and Zuckerman [37] consider the distribution obtained by using a bit-fixing source  $X = (X_1, \dots, X_n)$  to perform a random walk on a  $d$ -regular graph. (They consider a more general model of bit-fixing sources in which every symbol  $X_i$  ranges over an alphabet of size  $d$ ). The walk starts from some fixed vertex in the graph, and at step  $i$   $X_i$  is used to select a neighbor of the current vertex. They show that the distribution over the vertices converges to the uniform distribution at a rate which depends on  $k$  and the “spectral gap” of the graph. It is known that 2-regular graphs cannot have a small “spectral gap”. Indeed, this is why Kamp and Zuckerman consider alphabet size  $d > 2$ , which allows using  $d$ -regular expander graphs that have small spectral gap. Nevertheless,

---

<sup>2</sup>This was observed independently by Lipton and Vishnoi [40].

using their technique choosing the graph to be a short cycle of length  $k^{1/4}$  produces an extractor construction which extracts  $\log(k^{1/4}) = \Omega(\log k)$  bits.<sup>3</sup>

**A seeded extractor for bit-fixing sources with seed length  $O(\log \log n)$**

Converting the deterministic bit-fixing source extractor above into a seed obtainer we “obtain” an  $\Omega(\log k)$  bit seed. This allows us to use a seeded extractor with seed length  $d = \Omega(\log k)$ . However,  $d < \log n$  and by a lower bound of [48, 49] the class of high min-entropy sources does not have seeded extractors with seed  $d < \log n$ . To bypass this problem we construct a *seeded* extractor for *bit-fixing sources* with seed length  $O(\log \log n)$ . Note that the aforementioned deterministic extractor extracts these many bits as long as  $k > \log^c n$  for some constant  $c$  (when  $\Omega(\log k) \geq O(\log \log n)$ ).

The seeded extractor uses its seed to randomly partition the indices  $\{1, \dots, n\}$  into  $r$  sets  $T_1, \dots, T_r$  (for  $r$  equal, say to  $\log^4 n$ ), with the property that with high probability each one of these sets contains at least one good bit. We elaborate on this partitioning method later on. We then output  $r$  bits, where the  $i$ 'th bit is given by  $\oplus_{j \in T_i} x_j$ .

By combining the seed obtainer with the seeded bit-fixing source extractor we obtain a deterministic bit-fixing source extractor which extracts  $r = \log^4 n$  bits. To extract more bits, we convert this deterministic extractor into a seed obtainer. At this point we obtain a seed of length  $\log^4 n$  and can afford using a seeded extractor which extracts all the remaining randomness.

**Sampling and partitioning with only  $O(\log \log n)$  random bits**

We now explain how to use  $O(\log \log n)$  random bits to partition the indices  $\{1, \dots, n\}$  into  $r = \text{poly} \log n$  sets  $T_1, \dots, T_r$  such that for any set  $S \subseteq \{1, \dots, n\}$  of size  $k$ , with high probability (probability at least  $1 - O(1/\log n)$ ) all sets  $T_1, \dots, T_r$  contain approximately  $k/r$  indices from  $S$ .

Suppose we could afford using many random bits. A natural solution is to choose  $n$  random variables  $V_1, \dots, V_n \in \{1, \dots, r\}$  and have  $T_j$  be the set of indices  $i$  such that  $V_i = j$ . We expect  $k/r$  bits to fall in each  $T_j$  and by a union bound one can show that with high probability all sets  $T_1, \dots, T_r$  have a number of indices from  $S$  that is close to the expected value.

To reduce the number of random bits we derandomize the construction above and use random variables  $V_i$  which are  $\epsilon$ -close to being pairwise independent (for  $\epsilon = 1/\log^a n$  for some sufficiently large constant  $a$ ). Such variables can be constructed using only  $O(\log \log n)$  random bits [44, 2, 23] and suffice to guarantee the required properties.

The same technique also gives us a method for sampling a set  $T$  of indices in  $\{1, \dots, n\}$  (which we require in our construction of seed obtainers). We simply take the first set  $T_1$ . This sampling method uses only  $O(\log \log n)$  random bits and thus we can afford it when transforming our deterministic

---

<sup>3</sup>In fact, a similar idea is used in [37] in order to reduce the case of large  $d$  to the case of  $d = 2$ .

extractor into a seed obtainer. (Recall that our transformation uses part of the output of the deterministic extractor for sampling a subset of the indices). We remark that this sampling technique was used previously by Reingold et al. [56] as a component in a construction of seeded extractors.

### 2.1.4 Outline

In Section 2.2 we define the notations used in this chapter. In Section 2.3 we introduce the concept of seed obtainers and show how to construct them from deterministic bit-fixing source extractors and “averaging samplers”. In Section 2.4 we observe that the technique of [37] can be used to extract a few bits even when  $k$  is small. In Section 2.5 we give constructions for averaging samplers. In Section 2.6 we give a construction of a seeded bit-fixing source extractor that makes use of the sampling techniques of Section 2.5. In Section 2.7 we plug all the components together and prove our main theorems. Finally, in Section 2.8 we give some open problems.

## 2.2 Preliminaries

**Notations** We use  $[n]$  to denote the set  $\{1, \dots, n\}$ . We use  $P(S)$  to denote the set of subsets of a given set  $S$ . We use  $U_n$  to denote the uniform distribution over  $n$  bits. Given a distribution  $A$  we use  $w \leftarrow A$  to denote the experiment in which  $w$  is chosen randomly according to  $A$ . Given a string  $x \in \{0, 1\}^n$  and a set  $S \subseteq [n]$  we use  $x_S$  to denote the string obtained by restricting  $x$  to the indices in  $S$ . We denote the length of a string  $x$  by  $|x|$ . Logarithms will always be taken with base 2.

**Asymptotic notation** As this chapter has many parameters, we now explain exactly what we mean when using  $O(\cdot)$  and  $\Omega(\cdot)$  in a statement involving many parameters. We use the  $\Omega$  and  $O$  signs only to denote absolute constants (i.e., constants not dependent on any parameters even if these parameters are considered constants). Furthermore, when writing, for example,  $f(n) = O(g(n))$ , we always explicitly mention the conditions on  $n$  (and maybe other parameters) for which the statement holds.

### 2.2.1 Averaging Samplers

A sampler is a procedure which, given a short seed, generates a subset  $T \subseteq [n]$  such that for every set  $S \subseteq [n]$ ,  $|S \cap T|$  is with high probability “close to the expected size”.

**Definition 2.5.** An  $(n, k, k_{\min}, k_{\max}, \delta)$ -sampler  $\text{Samp} : \{0, 1\}^t \rightarrow P([n])$  is a function such that for any  $S \subseteq [n]$  such that  $|S| = k$ ,

$$\Pr_{w \leftarrow U_t} (k_{\min} \leq |\text{Samp}(w) \cap S| \leq k_{\max}) \geq 1 - \delta.$$



The definition above is nonstandard in several respects. In the more common definition (c.f. [30]), a sampler is required to work for sets of arbitrary size. In the definition above (which is similar in spirit to the one in [69]), the sampler is only required to work against sets of size  $k$  and the bounds  $k_{\min}, k_{\max}$  are allowed to depend on  $k$ . Furthermore, we require that the sampler have “distinct samples” as we do not allow  $T$  to be a multi-set.<sup>4</sup>

We will use samplers to “partition” bit-fixing sources. Note that in the case of an  $(n, k)$ -bit-fixing source, *Samp* returns a subset of indices such that, with high probability, the number of good bits in the subset is between  $k_{\min}$  and  $k_{\max}$ .

## 2.2.2 Probability Distributions

Some of the proofs in this chapter require careful manipulations of probability distributions. We use the following notation. We use  $U_m$  to denote the uniform distribution on  $m$  bit strings. We denote the probability of an event  $B$  under a probability distribution  $P$  by  $\Pr_P[B]$ . A random variable  $R$  that takes values in  $U$  is a function  $R : \Omega \rightarrow U$  (where  $\Omega$  is a probability space). We sometimes refer to  $R$  as a probability distribution over  $U$  (the distribution of the output of  $R$ ). For example, given a random variable  $R$  and a distribution  $P$  we sometimes write “ $R = P$ ”, and this means that the distribution of the output of  $R$  is equal to  $P$ . Given two random variables  $R_1, R_2$  over the same probability space  $\Omega$  we use  $(R_1, R_2)$  to denote the random variable induced by the function  $(R_1, R_2)(\omega) = (R_1(\omega), R_2(\omega))$ . Given two probability distributions  $P_1, P_2$  over domains  $\Omega_1, \Omega_2$ , we define  $P_1 \otimes P_2$  to be the product distribution of  $P_1$  and  $P_2$ , defined over the domain  $\Omega_1 \times \Omega_2$ .

**Definition 2.6** (conditioning distributions and random variables). *Given a probability distribution  $P$  over some domain  $U$  and an event  $A \subseteq U$  such that  $\Pr_P[A] > 0$  we define a distribution  $(P|A)$  over  $U$  as follows: Given an event  $B \subseteq U$ ,  $\Pr_{(P|A)}(B) = \Pr_P[B|A] = \frac{\Pr_P[A \cap B]}{\Pr_P[A]}$ .*

*We extend this definition to random variables  $R : \Omega \rightarrow U$ . Given an event  $A \subseteq \Omega$  we define  $(R|A)$  to be the probability distribution over  $U$  given by  $\Pr_{(R|A)}[B] = \Pr_R[R \in B|A]$ .*

We also need the notion of convex combination of distributions.

**Definition 2.7** (convex combination of distributions). *Given distributions  $P_1, \dots, P_t$  over  $U$  and coefficients  $\alpha_1, \dots, \alpha_t \geq 0$  such that  $\sum_{1 \leq i \leq t} \alpha_i = 1$ , we*

---

<sup>4</sup>We remark that some of the “standard techniques” for constructing averaging samplers (such as taking a walk on an expander graph or using a randomness extractor) perform poorly in this setup, and do not work when  $k < \sqrt{n}$  (even if  $T$  is allowed to be a multi-set). This happens because in order to even hit a set  $S$  of size  $k$ , these techniques require sampling a (multi-)set  $T$  of size larger than  $(n/k)^2$ , which is larger than  $n$  for  $k < \sqrt{n}$ . In contrast, note that a completely random set of size roughly  $n/k$  will hit a fixed set  $S$  of small size with high probability.

define the distribution  $P = \sum_{1 \leq i \leq t} \alpha_i P_i$  as follows: Given an event  $B \subseteq U$ ,  $\Pr_P[B] = \sum_{1 \leq i \leq t} \alpha_i \Pr_{P_i}[B]$ .

We also need the following technical lemmas.

**Lemma 2.8.** *Let  $X, Y$  and  $V$  be distributions over  $\{0, 1\}^n$  such that  $X$  is  $\epsilon$ -close to  $U_n$  and  $Y = \delta \cdot V + (1 - \delta) \cdot X$ . Then  $Y$  is  $(2\delta + \epsilon)$ -close to  $U_n$ .*

*Proof.* Let  $B \subseteq \{0, 1\}^n$  be some event.

$$\begin{aligned} |\Pr_Y(B) - \Pr_{U_n}(B)| &= |\delta \Pr_V(B) + (1 - \delta) \Pr_X(B) - \Pr_{U_n}(B)| \\ &\leq 2\delta + |\Pr_X(B) - \Pr_{U_n}(B)| \leq 2\delta + \epsilon. \end{aligned}$$

□

**Lemma 2.9.** *Let  $(A, B)$  be a random variable that takes values in  $\{0, 1\}^u \times \{0, 1\}^v$  and suppose that there exists some distribution  $P$  over  $\{0, 1\}^v$  such that for every  $a \in \{0, 1\}^u$  with  $\Pr[A = a] > 0$  the distribution  $(B|A = a)$  is  $\epsilon$ -close to  $P$ . Then  $(A, B)$  is  $\epsilon$ -close to  $(A \otimes P)$ .*

*Proof.*

$$\begin{aligned} &\frac{1}{2} \cdot \sum_{a,b} |\Pr[(A, B) = (a, b)] - \Pr_{A \otimes P}[a, b]| \\ &= \frac{1}{2} \cdot \sum_{a,b} |\Pr[A = a] \Pr[B = b|A = a] - \Pr[A = a] \Pr_P[b]| \\ &\leq \frac{1}{2} \cdot \sum_a \Pr[A = a] \sum_b |\Pr[B = b|A = a] - \Pr_P[b]| \leq \epsilon/2. \end{aligned}$$

□

**Lemma 2.10.** *Let  $(A, B)$  be a random variable that takes values in  $\{0, 1\}^u \times \{0, 1\}^v$  which is  $\epsilon$ -close to  $(A' \otimes U_v)$ ; then, for every  $b \in \{0, 1\}^v$  the distribution  $(A|B = b)$  is  $(\epsilon \cdot 2^{v+1})$ -close to  $A'$ .*

*Proof.* Assume for the purpose of contradiction that there exists some  $b^* \in \{0, 1\}^v$  such that the distribution  $(A|B = b^*)$  is not  $\alpha$ -close to  $A'$  for  $\alpha = \epsilon \cdot 2^{v+1}$ . Then there is an event  $D$  such that

$$|\Pr_{(A|B=b^*)}[D] - \Pr_{A'}[D]| > \alpha.$$

By complementing  $D$  if necessary, we can w.l.o.g. remove the absolute value from the inequality above. We define an event  $D'$  over  $\{0, 1\}^u \times \{0, 1\}^v$ . The event  $D' = \{(a, b)|b = b^*, a \in D\}$ . We have that

$$\Pr_{(A', U_v)}[D'] = \Pr_{A'}[D] \cdot 2^{-v}.$$

And similarly,

$$\Pr_{(A,B)}[D'] = \Pr_{(A|B=b^*)}[D] \Pr_B[B = b^*].$$

We know that  $B$  is  $\epsilon$ -close to  $U_v$ , and therefore  $\Pr_B[B = b^*] \geq 2^{-v} - \epsilon$ . Thus,

$$\begin{aligned} \Pr_{(A,B)}[D'] - \Pr_{(A',U_v)}[D'] &= \Pr_{(A|B=b^*)}[D] \Pr_B[B = b^*] - \Pr_{A'}[D] \cdot 2^{-v} \\ &\geq \Pr_{(A|B=b^*)}[D](2^{-v} - \epsilon) - \Pr_{A'}[D] \cdot 2^{-v} \geq 2^{-v} \left[ \Pr_{(A|B=b^*)}[D] - \Pr_{A'}[D] \right] - \epsilon. \end{aligned}$$

By our assumption the expression in square brackets is at least  $\alpha$ , and thus,

$$> 2^{-v} \alpha - \epsilon = \epsilon.$$

Thus, we get a contradiction.  $\square$

## 2.3 Obtaining an Independent Seed

### 2.3.1 Seed Obtainers and Their Application

One of the natural ways to try and extract *many* bits from imperfect random sources is to first run a “weak extractor” which extracts few bits from the input distribution and then use these few bits as a seed to a second extractor which extracts more bits. The obvious difficulty with this approach is that typically the output of the first extractor is *correlated* with the imperfect random source and it is not clear how to use it. (Seeded extractors are only guaranteed to work when the seed is *independent* from the random source). In the next definition we introduce the concept of a “seed obtainer” that overcomes this difficulty. Loosely speaking, a seed obtainer is a deterministic function which given a bit-fixing source  $X$  outputs a new bit-fixing source  $X'$  (with roughly the same randomness) together with a short random seed  $Y$  which is *independent* from  $X'$ . Thus, the seed  $Y$  can later be used to extract randomness from  $X'$  using a seeded extractor.

**Definition 2.11** (seed obtainer). *A function  $F : \{0, 1\}^n \rightarrow \{0, 1\}^n \times \{0, 1\}^d$  is a  $(k, k', \rho)$ -seed obtainer if for every  $(n, k)$ -bit-fixing source  $X$ , the distribution  $R = F(X)$  can be expressed as a convex combination of distributions  $R = \eta Q + \sum_a \alpha_a R_a$  (here the coefficients  $\eta$  and  $\alpha_a$  are nonnegative and  $\eta + \sum_a \alpha_a = 1$ ) such that  $\eta \leq \rho$  and for every  $a$  there exists an  $(n, k')$ -bit-fixing source  $Z_a$  such that  $R_a$  is  $\rho$ -close to  $Z_a \otimes U_d$ .*

It follows that given a seed obtainer one can use a *seeded extractor* for bit-fixing sources to construct a *deterministic* (i.e., seedless) extractor for bit-fixing sources.

**Theorem 2.3.** *Let  $F : \{0, 1\}^n \rightarrow \{0, 1\}^n \times \{0, 1\}^d$  be a  $(k, k', \rho)$ -seed obtainer. Let  $E_1 : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  be a seeded  $(k', \epsilon)$ -bit-fixing source extractor. Then  $E : \{0, 1\}^n \rightarrow \{0, 1\}^m$  defined by  $E(x) = E_1(F(x))$  is a deterministic  $(k, \epsilon + 3\rho)$ -bit-fixing source extractor*

*Proof.* By the definition of a seed obtainer we have that  $E(X) = \eta E_1(Q) + \sum_a \alpha_a E_1(R_a)$  for some  $\eta \leq \rho$ . For each  $a$  we have that  $E_1(R_a)$  is  $(\epsilon + \rho)$ -close to  $U_m$ . It follows that  $E(X)$  is  $(\epsilon + \rho)$ -close to  $\eta E_1(Q) + (1 - \eta)U_m$  and therefore by Lemma 2.8 we have that  $E(X)$  is  $(2\eta + \epsilon + \rho)$ -close to uniform. The lemma follows because  $2\eta + \epsilon + \rho \leq \epsilon + 3\rho$ .  $\square$

### 2.3.2 Constructing Seed Obtainers

Note that every seed obtainer “contains” a deterministic extractor for bit-fixing sources. More precisely, given a seed obtainer  $F(x) = (x', y)$  the function  $E(x) = y$  is a deterministic extractor for bit-fixing sources. We now show how to convert any deterministic bit-fixing source extractor with sufficiently small error into a seed obtainer.

Our construction appears in Figure 2.1. In words, given  $x$ , the seed obtainer first computes  $E(x)$ . It uses a part of  $E(x)$  as the second output  $y$  and another part to sample a substring of  $x$ . It obtains the first output  $x'$  by erasing the sampled substring from  $x$ . We now state the main theorem of this section.

**Theorem 2.4** (construction of seed obtainers). *For every  $n$  and  $k < n$ , Let  $Samp$  and  $E$  be as in Figure 2.1 (that is,  $Samp : \{0, 1\}^t \rightarrow P([n])$  is*

Figure 2.1: A seed obtainer for  $(n, k)$ -bit-fixing sources

#### Ingredients:

- An  $(n, k, k_{min}, k_{max}, \delta)$ -sampler  $Samp : \{0, 1\}^t \rightarrow P([n])$ .
- A deterministic  $(k_{min}, \epsilon)$ -bit-fixing source extractor  $E : \{0, 1\}^n \rightarrow \{0, 1\}^m$  with  $m > t$ .

**Result:** A  $(k, k', \rho)$ -seed obtainer  $F : \{0, 1\}^n \rightarrow \{0, 1\}^n \times \{0, 1\}^{m-t}$  with  $k' = k - k_{max}$  and  $\rho = \max(\epsilon + \delta, \epsilon \cdot 2^{t+1})$ .

#### The construction of $F$ :

- Given  $x \in \{0, 1\}^n$  compute  $E(x)$  and let  $E_1(x)$  denote the first  $t$  bits of  $E(x)$  and  $E_2(x)$  denote the remaining  $m - t$  bits.
- Let  $T = Samp(E_1(x))$ .
- Let  $x' = x_{[n] \setminus T}$ . If  $|x'| < n$  we pad it with zeroes to get an  $n$ -bit long string.
- Let  $y = E_2(x)$ , Output  $x', y$ .

an  $(n, k, k_{\min}, k_{\max}, \delta)$ -sampler and  $E : \{0, 1\}^n \rightarrow \{0, 1\}^m$  is a deterministic  $(k_{\min}, \epsilon)$ -bit-fixing source extractor). Then,  $F : \{0, 1\}^n \rightarrow \{0, 1\}^n \times \{0, 1\}^d$  defined in Figure 2.1 is a  $(k, k', \rho)$ -seed obtainer for  $d = m - t$ ,  $k' = k - k_{\max}$  and  $\rho = \max(\epsilon + \delta, \epsilon \cdot 2^{t+1})$ .

## Proof of Theorem 2.4

In this section we prove Theorem 2.4. Let  $E$  be a bit-fixing source extractor and  $Samp$  be a sampler which satisfy the requirements in Theorem 2.4. Let  $X$  be some  $(n, k)$ -bit-fixing source and let  $S \subseteq [n]$  be the set of  $k$  good indices for  $X$ . We will use capital letters to denote the random variables which come up in the construction. We split  $E(X)$  into two parts  $(E_1(X), E_2(X)) \in \{0, 1\}^t \times \{0, 1\}^{m-t}$ . For a string  $a \in \{0, 1\}^t$  we use  $T_a$  to denote  $Samp(a)$  and  $T'_a$  to denote  $[n] \setminus Samp(a)$ . Given a string  $x \in \{0, 1\}^n$ , we use  $x_a$  to denote  $x_{T_a}$  and  $x'_a$  to denote the  $n$  bit string obtained by padding  $x_{T'_a}$  to length  $n$ . Let  $X' = X'_{E_1(X)}$  and  $Y = E_2(X)$ . Our goal is to show that the pair  $(X', Y)$  is close to a convex combination of pairs of distributions where the first component is a bit-fixing source and the second is independent and uniformly distributed.

**Definition 2.12.** We say that a string  $a \in \{0, 1\}^t$  correctly splits  $X$  if  $k_{\min} \leq |S \cap T_a| \leq k_{\max}$ .

Note that by the properties of the sampler, almost all strings  $a$  correctly split  $X$ . We start by showing that for every *fixed*  $a$  which correctly splits  $X$  the variables  $X'_a$  and  $E(X)$  are essentially independent. Loosely speaking this happens because we can argue that there are enough good bits in  $X_a$  and therefore the extractor can extract randomness from  $X_a$  which is independent of the randomness in  $X'_a$ .

**Lemma 2.13.** For every fixed  $a \in \{0, 1\}^t$  which correctly splits  $X$  the pair of random variables  $(X'_a, E(X))$  is  $\epsilon$ -close to the pair  $(X'_a \otimes U_m)$ .

*Proof.* Let  $\ell = |Samp(a)|$ . Given a string  $\sigma \in \{0, 1\}^\ell$  and a string  $\sigma' \in \{0, 1\}^{n-\ell}$  we define  $[\sigma; \sigma']$  to be the  $n$  bit string obtained by placing  $\sigma$  in the indices of  $T_a$  and  $\sigma'$  in the indices of  $T'_a$ . More formally, we denote the  $\ell$  indices of  $T_a$  by  $i_1 < i_2 < \dots < i_\ell$  and the  $n - \ell$  indices of  $T'_a$  by  $i'_1 < i'_2 < \dots < i'_{n-\ell}$ . Given an  $i \in T_a$  we define  $index(i)$  to be the index  $j$  such that  $i_j = i$ , and equivalently given  $i \in T'_a$  we define  $index'(i)$  to be the index  $j$  such that  $i'_j = i$ . The string  $[\sigma; \sigma'] \in \{0, 1\}^n$  is defined as follows:

$$[\sigma; \sigma']_i = \begin{cases} \sigma_{index(i)} & i \in T_a \\ \sigma'_{index'(i)} & i \in T'_a \end{cases}$$

Note that in this notation  $X = [X_a; X'_a]$ . We are interested in the distribution of the random variable  $(X'_a, E(X)) = (X'_a, E([X_a; X'_a]))$ . For every

$b \in \{0, 1\}^{n-\ell}$  we consider the event  $\{X'_a = b\}$ . Fix some  $b \in \{0, 1\}^{n-\ell}$  such that  $\Pr[X'_a = b] > 0$ . The distribution

$$(E(X)|X'_a = b) = (E([X_a; X'_a])|X'_a = b) = E([X_a; b])$$

where the last equality follows because  $X_a$  and  $X'_a$  are independent and therefore  $X_a$  is not affected by fixing  $X'_a$ . Note that as  $a$  correctly splits  $X$ , the distribution  $[X_a; b]$  is a bit-fixing source with at least  $k_{\min}$  “good” bits. We conclude that for every  $b \in \{0, 1\}^{n-\ell}$  such that  $\Pr[X'_a = b] > 0$  the distribution  $(E(X)|X'_a = b)$  is  $\epsilon$ -close to uniform. We now apply Lemma 2.9 with  $A = X'_a$  and  $B = E(X)$  and conclude that the pair  $(X'_a, E(X))$  is  $\epsilon$ -close to  $(X'_a \otimes U_m)$ .  $\square$

We now argue that if  $\epsilon$  is small enough then the pair  $(X'_a, E_2(X))$  is essentially independent even when conditioning the probability space on the event  $\{E_1(X) = a\}$ .

**Lemma 2.14.** *For every fixed  $a \in \{0, 1\}^t$  that correctly splits  $X$ , the distribution  $((X'_a, E_2(X))|E_1(X) = a)$  is  $\epsilon \cdot 2^{t+1}$ -close to  $(X'_a \otimes U_{m-t})$ .*

*Proof.* First note that the statement is meaningless unless  $\epsilon < 2^{-t}$  we will assume w.l.o.g. that this is the case and then for every fixed  $a \in \{0, 1\}^t$  the event  $\{E_1(X) = a\}$  occurs with nonzero probability as  $E_1(X)$  is  $\epsilon$ -close to uniform over  $\{0, 1\}^t$ . The lemma will follow as a straightforward application of Lemma 2.10. We set  $A = (X'_a, E_2(X))$ ,  $B = E_1(X)$  and  $A' = (X'_a, U_{m-t})$ . We indeed have that  $(A, B)$  is  $\epsilon$ -close to  $(A', U_t)$  and the lemma follows.  $\square$

We are now ready to prove Theorem 2.4.

*Proof.* (of Theorem 2.4) By the properties of the extractor we have that  $E_1(X)$  is  $\epsilon$ -close to uniform. It follows (by the properties of the sampler) that the probability that  $E_1(X)$  correctly splits  $X$  is  $1 - \eta$  for some  $\eta$  which satisfies  $\eta \leq \epsilon + \delta$ . We now consider the output random variable  $R = (X', E_2(X))$ . We need to express this random variable as a convex combination of independent distributions and a small error term. We set  $Q$  to be the distribution  $(R|E_1(X) \text{ doesn't correctly split } X)$ . For every correctly splitting  $a$  we set  $R_a$  to be the distribution  $(R|E_1(X) = a)$  and  $\alpha_a = \Pr[E_1(X) = a]$ . By our definition we have that indeed  $R = \eta Q + \sum_a \alpha_a R_a$ . For every  $a$  that correctly splits  $X$  we have that  $R_a = ((X', E_2(X))|E_1(X) = a) = ((X'_{E_1(X)}, E_2(X))|E_1(X) = a) = ((X'_a, E_2(X))|E_1(X) = a)$ . By Lemma 2.14 we have that  $R_a$  is  $\epsilon \cdot 2^{t+1}$ -close to  $(X'_a \otimes U_{m-t})$ . As  $a$  correctly splits  $X$  we have that  $X'_a$  is an  $(n, k - k_{\max})$ -bit-fixing source as required. Thus, we have shown that the distribution  $R_a$  is close to a convex combination of pairs of essentially independent distributions where the first is a bit-fixing source and the second is uniform.  $\square$

## 2.4 Extracting a Few Bits for Any $k$

The deterministic bit-fixing source extractor of Kamp and Zuckerman [37] only works for  $k > \sqrt{n}$ . However, their technique easily gives a deterministic bit-fixing source extractor that extracts very few bits ( $\Omega(\log k)$  bits) from a bit-fixing source with arbitrarily small  $k$ . We will later use this extractor to construct a seed obtainer that will enable us to extract many more bits.

**Theorem 2.5.** *For every  $n > k \geq 100$  there is an explicit deterministic  $(k, 2^{-\sqrt{k}})$ -bit-fixing source extractor  $E : \{0, 1\}^n \rightarrow \{0, 1\}^{(\log k)/4}$ .*

For the proof, we need the following result, which is a very special case of Lemma 3.3 in [37].

**Lemma 2.15.** *([37, Lemma 3.3] for  $\epsilon = 0$  and  $d = 2$ ). Let the graph  $G$  be an odd cycle with  $M$  vertices and second eigenvalue  $\lambda$ . Suppose we take a walk on  $G$  for  $n$  steps, starting from some fixed vertex  $v$  with the steps taken according to the symbols from an  $(n, k)$ -bit-fixing source  $X$ . Let  $Z$  be the distribution on the vertices at the end of the walk; then  $Z$  is  $\left(\frac{1}{2}\lambda^k\sqrt{M}\right)$ -close to the uniform distribution on  $[M]$ .*

To extract a few bits from a bit-fixing source  $X$ , we will use the bits of  $X$  to conduct a random walk on a small cycle.

*Proof.* (of Theorem 2.5) We use the source-string to take a walk on a cycle of size  $\sqrt[4]{k}$  from a fixed vertex. The second eigenvalue of a  $d$ -cycle is  $\cos(\frac{\pi}{d})$  ([41, Ex. 11.1]). Using Lemma 2.15, we reach distance  $\left(\cos\left(\frac{\pi}{\sqrt[4]{k}}\right)\right)^k k^{1/8}$  from uniform. By the Taylor expansion of  $\cos$ , for  $0 < x < 1$

$$\cos(x) < 1 - \frac{x^2}{2} + \frac{x^4}{24} < 1 - \frac{x^2}{4}$$

Therefore

$$\begin{aligned} \left(\cos\left(\frac{\pi}{\sqrt[4]{k}}\right)\right)^k &< \left(1 - \frac{\pi^2}{4\sqrt{k}}\right)^k \\ &< \left(e^{-\frac{\pi^2}{4}}\right)^{\sqrt{k}} < 4^{-\sqrt{k}} \end{aligned}$$

where the second to last inequality holds because  $(1-x) < e^{-x}$  for  $0 < x < 1$ . Therefore, we reach distance  $4^{-\sqrt{k}}k^{1/8} \leq 2^{-\sqrt{k}}$ . By outputting the final vertex's name we get  $\frac{\log(k)}{4}$  bits with the same distance from uniform.  $\square$

## 2.5 Sampling and Partitioning with a Short Seed

Let  $S \subseteq [n]$  be some subset of size  $k$ . In this section we show how to use few random bits in order perform two related tasks.

**Sampling:** Generate a subset  $T \subseteq [n]$  such that  $|S \cap T|$  is in a prespecified interval  $[k_{\min}, k_{\max}]$  (see Definition 2.5).

**Partitioning:** Partition  $[n]$  into  $r$  distinct subsets  $T_1, \dots, T_r$  such that for every  $1 \leq i \leq r$ ,  $|S \cap T_i|$  is in a prespecified interval  $[k_{\min}, k_{\max}]$ . Needless to say, a partitioning scheme immediately implies a sampling scheme by concentrating on a single  $T_i$ .

In this section we present two constructions of such schemes. The first construction is used in our deterministic bit-fixing source extractor for  $k > \sqrt{n}$ . In this setup we can allow the sampler to use many random bits (say  $n^{\Omega(1)}$  bits) and can have error  $2^{-n^{\Omega(1)}}$ .

**Lemma 2.16** (sampling with low error). *Fix any constants  $0 < \gamma \leq 1/2$  and  $\alpha > 0$ . There exists a constant  $n'$  depending on  $\alpha$  and  $\gamma$  such that for any integers  $n, k$  satisfying  $n > n'$  and  $n^{1/2+\gamma} \leq k \leq n$ , there exists an  $(n, k, (n^{1/2+\gamma})/6, n^{1/2+\gamma}, 2^{-\Omega(\alpha \cdot n^\gamma)})$ -sampler  $\text{Samp} : \{0, 1\}^t \rightarrow P([n])$  where  $t = \alpha \cdot n^{2\gamma}$ .*

The second construction is used in our deterministic bit-fixing source extractor for small  $k$ . For that construction we require schemes that use only  $\alpha \log k$  bits for some small constant  $\alpha > 0$ . The construction of Lemma 2.16 requires at least  $\log n > \log k$  bits, which is too much. Instead, we use a different construction which has much larger error (e.g.,  $k^{-\Omega(1)}$ ).

**Lemma 2.17** (sampling with  $O(\log k)$  bits). *Fix any constant  $0 < \alpha < 1$ . There exist constants  $c > 0, 0 < b < 1$  and  $1/2 < e < 1$  (all depending on  $\alpha$ ) such that for any  $n \geq 16$  and  $k \geq \log^c n$ , we obtain an explicit  $(n, k, k^e / 2, 3 \cdot k^e, O(k^{-b}))$ -sampler  $\text{Samp} : \{0, 1\}^t \rightarrow P([n])$  where  $t = \alpha \cdot \log k$ .*

**Lemma 2.18** (partitioning with  $O(\log k)$  bits). *Fix any constant  $0 < \alpha < 1$ . There exist constants  $c > 0, 0 < b < 1$  and  $1/2 < e < 1$  (all depending on  $\alpha$ ) such that for any  $n \geq 16$  and  $k \geq \log^c n$ , we can use  $\alpha \cdot \log k$  random bits to explicitly partition  $[n]$  into  $m = \Omega(k^b)$  sets  $T_1, \dots, T_m$  such that for any  $S \subseteq [n]$  where  $|S| = k$*

$$\Pr(\forall i, \quad k^e/2 \leq |T_i \cap S| \leq 3 \cdot k^e) \geq 1 - O(k^{-b}).$$

The first construction is based on “ $\ell$ -wise independence”, and the second is based on “almost 2-wise dependence” [44, 2, 23]. Sampling techniques based on  $\ell$ -wise independence were first suggested by Bellare and Rompel [6]. However, this technique is not good enough in our setting and we use a different approach (which was also used in [56] with slightly different parameters). In Appendix A we explain the approach in detail, compare it to the approach of [6] and give full proofs of the lemmas above.



## 2.6 A Seeded Bit-Fixing Source Extractor with a Short Seed

In this section we give a construction of a seeded bit-fixing source extractor that uses seed length  $O(\log k)$  to extract  $k^{\Omega(1)}$  bits as long as  $k$  is not too small. This seeded extractor is used as a component in our construction of deterministic extractors for bit-fixing sources.

**Theorem 2.6.** *Fix any constant  $0 < \alpha < 1$ . There exist constants  $c > 0$  and  $0 < b < 1$  (both depending on  $\alpha$ ) such that for any  $n \geq 16$  and  $k \geq \log^c n$ , there exists an explicit seeded  $(k, \epsilon)$ -bit-fixing source extractor  $E : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  with  $d = \alpha \cdot \log k$ ,  $m = \Omega(k^b)$  and  $\epsilon = O(k^{-b})$ .*

*Proof.* Let  $X$  be an  $(n, k)$ -bit-fixing source. Let  $x = x_1, \dots, x_n$  be a string sampled by  $X$ . The extractor  $E$  works as follows: We use the extractor seed  $y$  to construct a partition of the bits of  $x$  into  $m$  sets. Then we output the xor of the bits in each set. With high probability, each set will contain a good bit and therefore, with high probability, the output will be uniformly distributed.

More formally, let  $b$  and  $c$  be the constants from Lemma 2.18 when using the lemma with the parameter  $\alpha$ .

### E(x,y)

- We use the seed  $y$  to obtain a partition of  $[n]$  into  $m = \Omega(k^b)$  sets  $T_1, \dots, T_m$  using Lemma 2.18 with the parameter  $\alpha$ .
- For  $1 \leq i \leq m$ , compute  $z_i = \bigoplus_{j \in T_i} x_j$ .
- Output  $z = z_1, \dots, z_m$ .

We give a detailed correctness proof although it is very straightforward: Let  $S \subseteq [n]$  be the set of good indices and let  $Z$  be the distribution of the output string  $z$ . We need to prove that  $Z$  is close to uniform. Let  $A$  be the event  $\{\forall i \ T_i \cap S \neq \emptyset\}$ . That is,  $A$  is the "good" event in which all sets contain a random bit (and therefore in this case the output is uniform). Let  $A^c$  be the complement event, i.e.,  $A^c$  is the event  $\{\exists i \ T_i \cap S = \emptyset\}$ . We decompose  $Z$  according to  $A$  and  $A^c$ :

$$Z = \Pr(A^c) \cdot (Z|A^c) + \Pr(A) \cdot (Z|A)$$

$(Z|A)$  is uniformly distributed. From Lemma 2.18, when  $k \geq \log^c n$ ,  $\Pr(A) \geq 1 - O(k^{-b})$ . Therefore, by Lemma 2.8

$$Z \stackrel{O(k^{-b})}{\sim} U_m.$$

□

## 2.7 Deterministic Extractors for Bit-Fixing Sources

In this section, we compose the ingredients from previous sections to prove Theorems 2.1 and 2.2. Namely, given choices for a deterministic bit-fixing source extractor, a sampler and a seeded bit-fixing source extractor, we use Theorems 2.3 and 2.4 to get a new deterministic bit-fixing source extractor. This works as follows: We “plug in” a deterministic extractor that extracts a little randomness and a sampler into Theorem 2.4 to get a seed obtainer. We then “plug in” this seed obtainer and a seeded extractor into Theorem 2.3 to get a new deterministic extractor which extracts almost all of the randomness. It is convenient to express this composition as follows:

**Theorem 2.7.** *Assume we have the following ingredients:*

- An  $(n, k, k_{\min}, k_{\max}, \delta)$ -sampler  $Samp : \{0, 1\}^t \rightarrow P([n])$ ,
- a deterministic  $(k_{\min}, \epsilon^*)$ -bit-fixing source extractor  $E^* : \{0, 1\}^n \rightarrow \{0, 1\}^{m'}$  and
- a seeded  $(k - k_{\max}, \epsilon_1)$ -bit-fixing source extractor  $E_1 : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$

where  $m' \geq d + t$ . Then we construct a deterministic  $(k, \epsilon)$ -bit-fixing source extractor  $E : \{0, 1\}^n \rightarrow \{0, 1\}^m$  where  $\epsilon = \epsilon_1 + 3 \cdot \max(\epsilon^* + \delta, \epsilon^* \cdot 2^{t+1})$ .

*Proof.* We use  $Samp$  and  $E^*$  in Theorem 2.4 to get a  $(k, k - k_{\max}, \max(\epsilon^* + \delta, \epsilon^* \cdot 2^{t+1}))$ -seed obtainer  $F : \{0, 1\}^n \rightarrow \{0, 1\}^n \times \{0, 1\}^{m'-t}$ . Since  $m' - t \geq d$ , we can use  $F$  and  $E_1$  in Theorem 2.3 to obtain a deterministic  $(k, \epsilon)$ -bit-fixing source extractor  $E : \{0, 1\}^n \rightarrow \{0, 1\}^m$  where  $\epsilon = \epsilon_1 + 3 \cdot \max(\epsilon^* + \delta, \epsilon^* \cdot 2^{t+1})$ .  $\square$

We also require the following construction of a seeded extractor (which is in particular a seeded bit-fixing source extractor).

**Theorem 2.8** ([55]). *For any  $n, k$  and  $\epsilon > 0$ , there exists a  $(k, \epsilon)$ -extractor  $Ext : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  where  $m = k$  and  $d = O(\log^2 n \cdot \log(1/\epsilon) \cdot \log k)$*

### 2.7.1 An Extractor for Large $k$ (Proof of Theorem 2.1)

To prove Theorem 2.1, we first state results about the required ingredients and then use the ingredients in Theorem 2.7.

We use the deterministic bit-fixing source extractor of Kamp and Zuckerman [37]. Loosely speaking, the following theorem states that when  $k \gg \sqrt{n}$ , we can deterministically extract a polynomial fraction of the randomness with an exponentially small error.

**Theorem 2.9** ([37]). *Fix any integers  $n, k$  such that  $k = b \cdot n^{1/2+\gamma}$  for some  $b > 0$  and  $0 < \gamma \leq 1/2$ . There exists a constant  $c > 0$  (not depending on any of the parameters) such that there exists an explicit deterministic  $(k, \epsilon^*)$ -bit-fixing source extractor  $E^* : \{0, 1\}^n \rightarrow \{0, 1\}^m$  where  $m = cb^2 \cdot n^{2\gamma}$  and  $\epsilon^* = 2^{-m}$ .*

Using the theorem above we can obtain a seed of length  $O(n^{2\gamma})$ . This means that we can afford these many bits for our sampler and seeded bit-fixing source extractor. We use the sampler based on  $\ell$ -wise independence from Lemma 2.16. We use the seeded extractor of [55] (Theorem 2.8), which we now restate in the following form:

**Corollary 2.10.** *Fix any constants  $0 < \gamma \leq 1/2$  and  $\alpha > 0$ . There exists a constant  $n'$  depending on  $\gamma$  such that for any integers  $n, k$  satisfying  $n > n'$  and  $k \leq n$  there exists a  $(k, \epsilon_1)$ -extractor  $E_1 : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  where  $m = k$ ,  $d = \alpha \cdot n^{2\gamma}$  and  $\epsilon_1 = 2^{-\Omega(\alpha \cdot n^\gamma)}$ .*

*Proof.* We use the extractor of Theorem 2.8. We need  $d = c_1 \cdot (\log^3 n \cdot \log(1/\epsilon_1))$  random bits for some constant  $c_1 > 0$ . We want to use at most  $\alpha \cdot n^{2\gamma}$  random bits. We get the inequality  $\alpha \cdot n^{2\gamma} \geq c_1 \cdot \log^3 n \cdot \log(1/\epsilon_1)$ . Equivalently,  $\epsilon_1 \geq 2^{-\frac{\alpha \cdot n^{2\gamma}}{c_1 \cdot \log^3 n}}$ . So for a large enough  $n$  (depending on  $\gamma$ ), we can take  $\epsilon_1 = 2^{-\frac{\alpha \cdot n^\gamma}{c_1}} = 2^{-\Omega(\alpha \cdot n^\gamma)}$ .  $\square$

We now compose the ingredients from Theorem 2.9, Lemma 2.16 and Corollary 2.10 to prove Theorem 2.1. The composition is a bit cumbersome in terms of the different parameters. The main issue is that when  $k = n^{1/2+\gamma}$ , the deterministic extractor of Kamp and Zuckerman extracts  $\Omega(n^{2\gamma})$  random bits; and this is enough to use as a seed for a sampler and seeded extractor (that extracts all the randomness) with error  $2^{-\Omega(n^\gamma)}$ .

*Proof.* (of Theorem 2.1) Let  $c$  be the constant in Theorem 2.9. We use Theorem 2.7 with the following ingredients:

- The  $(n, k, (n^{1/2+\gamma})/6, n^{1/2+\gamma}, \delta = 2^{-\Omega(n^\gamma)})$ -sampler  $Samp : \{0, 1\}^t \rightarrow P([n])$  from Lemma 2.16 where  $t = (c/72)n^{2\gamma}$ .
- The deterministic  $((n^{1/2+\gamma})/6, \epsilon^* = 2^{-m'})$ -bit-fixing source extractor  $E^* : \{0, 1\}^n \rightarrow \{0, 1\}^{m'}$  from Theorem 2.9 where  $m' = (c/36)n^{2\gamma}$ .
- The  $(k - n^{1/2+\gamma}, \epsilon_1 = 2^{-\Omega(n^\gamma)})$ -extractor  $E_1 : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  from Corollary 2.10 with  $d \leq (c/72)n^{2\gamma}$  and  $m = k - n^{1/2+\gamma}$ .

Note that all three objects exist for a large enough  $n$  depending only on  $\gamma$  ( $c$  is a universal constant). Note that  $m' \geq t + d$ . Therefore, applying Theorem 2.7, we get a deterministic  $(k, \epsilon)$ -bit-fixing source extractor  $E : \{0, 1\}^n \rightarrow \{0, 1\}^m$  where  $m = k - n^{1/2+\gamma}$  and

$$\epsilon = \epsilon_1 + 3 \cdot \max(\epsilon^* + \delta, \epsilon^* \cdot 2^{t+1})$$

$$\begin{aligned}
&= 2^{-\Omega(n^\gamma)} + 3 \cdot \max \left( 2^{-(c/36)n^{2\gamma}} + 2^{-\Omega(n^\gamma)}, 2^{-(c/36)n^{2\gamma}} \cdot 2^{(c/72)n^{2\gamma}+1} \right) \\
&= 2^{-\Omega(n^\gamma)} + 3 \cdot \max \left( 2^{-\Omega(n^\gamma)}, 2^{-(c/72)n^{2\gamma}+1} \right) = 2^{-\Omega(n^\gamma)}
\end{aligned}$$

(for a large enough  $n$  depending on  $\gamma$ ).  $\square$

### 2.7.2 An Extractor for Small $k$ (Proof of Theorem 2.2)

To prove Theorem 2.2 we need a deterministic bit-fixing source extractor for  $k < \sqrt{n}$ . We use the extractor of Theorem 2.5. We prove the theorem in two steps. First, we use Theorem 2.7 to convert the initial extractor into a deterministic bit-fixing source extractor that extracts more bits. We then apply Theorem 2.7 again to obtain a deterministic bit-fixing source extractor that extracts almost all bits.

The following lemma implements the first step and shows how to extract a polynomial fraction of the randomness with a polynomially small error whenever  $k \geq \log^c n$  for some constant  $c$ .

**Lemma 2.19.** *There exist constants  $c, b > 0$  such that for any  $k \geq \log^c n$  and large enough  $n$ , there exists an explicit deterministic  $(k, k^{-b})$ -bit-fixing source extractor  $E : \{0, 1\}^n \rightarrow \{0, 1\}^m$  where  $m = k^{\Omega(1)}$ .*

*Proof.* Roughly speaking, the main issue is that we can get  $\Omega(\log k)$  random bits using the deterministic extractor of Theorem 2.5. We will need  $c_1 \cdot \log \log n$  random bits to use the sampler of Lemma 2.17 and the seeded extractor of Theorem 2.6 (for some constant  $c_1$ ). Thus, when  $k \geq \log^c n$  for large enough  $c$ , we will have enough bits.

Formally, we use Theorem 2.7 with the following ingredients:

- The  $(n, k, k^e/2, 3 \cdot k^e, \delta = k^{-\Omega(1)})$ -sampler  $Samp : \{0, 1\}^t \rightarrow P([n])$  from Lemma 2.17 where  $t = \log k/32$  and  $e > 1/2$  is the constant from that lemma.
- The deterministic  $(k^e/2, \epsilon^* = 2^{-\sqrt{k^e/2}})$ -bit-fixing source extractor  $E^* : \{0, 1\}^n \rightarrow \{0, 1\}^{m'}$  from Theorem 2.5 where  $m' = \log(k^e/2)/4$ .
- The seeded  $(k - 3 \cdot k^e, \epsilon_1 = (k - 3 \cdot k^e)^{-\Omega(1)})$ -bit-fixing source extractor  $E_1 : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  from Theorem 2.6 with  $d = \log k/32$  and  $m = (k - 3 \cdot k^e)^{\Omega(1)}$ .

Note that all three objects exist for  $k \geq \log^c n$  for some constant  $c$  and large enough  $n$ . Assume that  $n$  is large enough so that  $k \geq \log^c n \geq 2$ . To use Theorem 2.7 we need to check that  $m' \geq t + d$ : Indeed,  $m' = \log(k^e/2)/4 \geq \log k/16 = t + d$  (where we used  $e > 1/2$ , as stated in Lemma 2.17). Applying Theorem 2.7, we get a deterministic  $(k, \epsilon)$ -bit-fixing source extractor  $E : \{0, 1\}^n \rightarrow \{0, 1\}^m$ . Notice that for large enough  $n$ ,  $\epsilon_1 = k^{-\Omega(1)}$ ; therefore

$$\epsilon = \epsilon_1 + 3 \cdot \max(\epsilon^* + \delta, \epsilon^* \cdot 2^{t+1})$$

$$= k^{-\Omega(1)} + 3 \cdot \max \left( 2^{-\sqrt{k^e/2}} + k^{-\Omega(1)}, 2^{-\sqrt{k^e/2}} \cdot 2^{\log k/32+1} \right) = k^{-\Omega(1)}$$

(for a large enough  $n$ ). Also,  $m = (k - 3 \cdot k^e)^{\Omega(1)} = k^{\Omega(1)}$  (for a large enough  $n$ ) so we get the required parameters.  $\square$

We now compose the ingredients from Lemmas 2.17 and 2.19 and Theorem 2.8 to prove Theorem 2.2. The composition is a bit cumbersome in terms of the different parameters. The main issue is that we can extract  $k^{\Omega(1)}$  random bits using the deterministic extractor of Lemma 2.19. We want  $\log^5 n$  random bits to use the seeded extractor of Theorem 2.8. Thus, when  $k \geq \log^c n$  for large enough  $c$ , we will have enough bits.

*Proof.* (of Theorem 2.2) Let  $b$  be the constant in Lemma 2.19. We use Theorem 2.7 with the following ingredients:

- The  $(n, k, k^e/2, 3 \cdot k^e, \delta = k^{-\Omega(1)})$ -sampler  $\text{Samp} : \{0, 1\}^t \rightarrow P([n])$  from Lemma 2.17 where  $t = (b/2) \log k$  and  $e > 1/2$  is the constant from that lemma.
- The deterministic  $(k^e/2, \epsilon^* = (k^e/2)^{-b})$ -bit-fixing source extractor  $E^* : \{0, 1\}^n \rightarrow \{0, 1\}^{m'}$  from Lemma 2.19 where  $m' = (k^e/2)^{\Omega(1)}$ .
- The  $(k - 3 \cdot k^e, \epsilon_1 = 1/n)$ -extractor  $E_1 : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  from Theorem 2.8 with  $d \leq \log^5 n$  and  $m = (k - 3 \cdot k^e)$ .

Note that all three objects exist for  $k \geq \log^c n$  for some constant  $c$  and for large enough  $n$ . To use Theorem 2.7 we need to check that  $m' \geq t + d$ ; note that  $m' = k^{\Omega(1)}$ . We take  $c$  large enough so that for large enough  $n$   $m'/2 > \log^5 n$  and  $m'/2 > (b/2)/\log k$ . So for such  $n$

$$m' \geq \log^5 n + (b/2) \log k \geq d + t.$$

Applying Theorem 2.7, we get a deterministic  $(k, \epsilon)$ -bit-fixing source extractor  $E : \{0, 1\}^n \rightarrow \{0, 1\}^m$ , where

$$\begin{aligned} \epsilon &= \epsilon_1 + 3 \cdot \max(\epsilon^* + \delta, \epsilon^* \cdot 2^{t+1}) \\ &= 1/n + 3 \cdot \max \left( (k^e/2)^{-b} + k^{-\Omega(1)}, 2 \cdot (k^e/2)^{-b} \cdot k^{b/2} \right) = k^{-\Omega(1)} \end{aligned}$$

(for large enough  $n$ ). Since  $m = k - O(k^e)$  where  $1/2 < e < 1$  we are done.  $\square$

## 2.8 Discussion and Open Problems

We give explicit constructions of deterministic bit-fixing source extractors that extract almost all the randomness. However, we achieve rather large error  $\epsilon = k^{-\Omega(1)}$  in the case where  $k < \sqrt{n}$ . We now explain why this

happens and suggest how to reduce the error. Recall that in this case our final extractor is based on an initial extractor that extracts only  $m = O(\log k)$  bits. When transforming the initial extractor into the final extractor we use the output bits of the initial extractor as a seed for an averaging sampler. The error parameter  $\delta$  of an averaging sampler has to be larger than  $2^{-m}$ , and as this error is “inherited” by the final extractor we can only get error about  $1/k$ . A natural way to improve our result is to find a better construction for the initial extractor.

Some applications of deterministic bit-fixing source extractors in adaptive settings of exposure-resilient cryptography require extractors with  $\epsilon \ll 2^{-m}$ . We do not achieve this goal (even in our first construction that has relatively small error (unless we artificially shorten the output)). Suppose one wants to extract  $m = k - u$  bits (for some parameter  $u$ ). It is interesting to investigate how small the error can be as a function of  $u$ ? We point out that the existential nonexplicit result achieves error  $\epsilon \geq 2^{-u}$  and thus cannot achieve  $\epsilon < 2^{-m}$  when  $m \geq k/2$ . We remark that for bit-fixing sources we have examples of settings where the nonexplicit result is not optimal. For example, when  $m = 1$  the xor-extractor is errorless (see also [15]). Given the discussion above we find it interesting to achieve  $m = \Omega(k)$  with  $\epsilon = 2^{-\Omega(k)}$  for every choice of  $k$ .

Deterministic Extraction from Weak Random Sources

Gabizon, A.

2011, XII, 148 p., Hardcover

ISBN: 978-3-642-14902-3