Deterministic extractors for bit-fixing sources by obtaining an independent seed

Ariel Gabizon[*] Weizmann Institute

Ran Raz[†] Weizmann Institute

Ronen Shaltiel[‡] Weizmann Institute

Abstract

An (n, k)-bit-fixing source is a distribution X over $\{0,1\}^n$ such that there is a subset of k variables in $X_1, \ldots, X_n$ 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 $E : \{0,1\}^n \to \{0,1\}^m$ which on an arbitrary (n, k)-bit-fixing source outputs m bits that are statistically-close to uniform. Recently, Kamp and Zuckerman [13] gave a construction of deterministic bit-fixing source extractor that extracts $\Omega(k^2/n)$ bits, and requires $k > \sqrt{n}$.

In this paper 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.

1. Introduction

1.1. Deterministic randomness extractors

A "deterministic randomness extractor" is a function that "extracts" bits that are (statistically close) to uniform from "weak sources of randomness" which may be very far from uniform.

Definition 1.1 (deterministic extractor). *Let $\mathcal{C}$ be a class of distributions on $\{0,1\}^n$. A function $E : \{0,1\}^n \to \{0,1\}^m$ is a deterministic ϵ-extractor for $\mathcal{C}$ if for every distribution $X \in \mathcal{C}$ the distribution $E(X)$ (obtained by sampling x from X and computing $E(x)$) is ϵ-close to the uniform distribution on m bit strings.*[1]

The distributions $X \in \mathcal{C}$ are often referred to as "weak random sources". That is, distributions that "contain" some randomness. Given a class $\mathcal{C}$ the goal of this field is to design *explicit* (that is efficiently computable) deterministic extractors that extract as many random bits as possible.

1.2. Some related work on randomness extraction.

Various classes $\mathcal{C}$ of distributions were studied in the literature: The first construction of deterministic extractors can be traced back to von Neumann [30] who showed how to use many independent tosses of a biassed coin (with unknown bias) to obtain an unbiased coin. Blum [4] considered sources that are generated by a walk on a finite Markov-chain. Santha and Vazirani [23], Vazirani [27, 28], Chor and Goldreich [7] and Barak et al. [2] studied sources that are composed of several independent samples from various classes of distributions. Trevisan and Vadhan [25] studied sources which are "samplable" by small circuits.

A negative result was given by Santha and Vazirani that exhibit a very natural class of high-entropy sources that does not have deterministic extractors. This led to the development of a different notion of extractors called "seeded extractors". Such extractors are allowed to use a short seed of few truly random bits when extracting randomness from a source. (The notion of "seeded extractors" emerged from attempts to simulate probabilistic algorithms using weak random sources [29, 7, 9, 31, 32] and was explicitly defined by Nisan and Zuckerman [18].) Unlike deterministic extractors, seeded extractors can extract randomness from the most general class of sources: Sources with high (min)-entropy. The reader is referred to [16, 17, 24, 26] for various surveys on randomness extractors.

1.3. Bit-fixing sources

In this paper we concentrate on the family of "bit-fixing sources" introduced in [8]. A distribution X over $\{0,1\}^n$ is

∗ Research supported by Israel Science Foundation (ISF) grant.

† Research supported by Israel Science Foundation (ISF) grant.

‡ Research supported by the Koshland scholarship.

1 Two distributions P and Q over $\{0,1\}^m$ are ϵ-close (denoted by $P \overset{\epsilon}{\sim} Q$) if for every event $A \subseteq \{0,1\}^m$, $|P(A) - Q(A)| \leq \epsilon$.

a bit-fixing source if there is a subset $S \subseteq \{1, \ldots, n\}$ 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.[2]

Definition 1.2 (bit-fixing sources and deterministic extractors). *A distribution X over $\{0,1\}^n$ is an (n,k)-bit-fixing source if there exists a subset $S = \{i_1, \cdots, i_k\} \subseteq \{1, \ldots, n\}$ such that $X_{i_1}, X_{i_2}, \cdots, X_{i_k}$ is uniformly distributed over $\{0,1\}^k$ and for every $i \notin S$, X_i is constant.*

A function $E : \{0,1\}^n \to \{0,1\}^m$ is a deterministic (k,ϵ)-bit-fixing source extractor if it is a deterministic ϵ-extractor 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 [8]. 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 [22, 5, 6, 11]. The reader is referred to [10] 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 $(k, 1/3)$-bit-fixing source extractor can extract at most k bits. The function $E(x) = \oplus_{1 \le i \le n} x_i$ is a deterministic $(k,0)$-bit-fixing source extractor which extracts one bit for any $k \ge 1$. Chor et al. [8] concentrated on "errorless" extractors (that is 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 errorless extractors for large k. A probabilistic argument shows the existence of a deterministic (k,ϵ)-bit-fixing source extractor that extracts $m = k - O(\log(n/\epsilon))$ bits for any choice of k and ϵ. Thus, it is natural to try and achieve such parameters by explicit constructions.

In a recent paper Kamp and Zuckerman [13] constructed explicit deterministic (k,ϵ)-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 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 $0 < \alpha \le 1/2$ the extractor only extracts $m = \eta n^{2\alpha}$ bits.

2 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. In this paper we are only concerned with the "oblivious case".

1.4. Our results

In this paper, 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 1.3. *For every constant $\gamma > 0$ and for any k, there is an explicit deterministic (k,ϵ)-bit-fixing source extractor $E : \{0,1\}^n \to \{0,1\}^m$ where $m = k - n^{1/2+\gamma}$ and $\epsilon = 2^{-\Omega(n^\gamma)}$.*

Consider $k = n^{1/2+\alpha}$ for some constant $\alpha > 0$. We can choose any $\gamma < \alpha$ and extract $m = n^{1/2+\alpha} - n^{1/2+\gamma}$ bits whereas the construction of [13] only extracts $m = O(n^{2\alpha})$ bits. For this choice of parameters we achieve error $\epsilon = 2^{-\Omega(n^\gamma)}$ whereas [13] achieves a slightly smaller error $\epsilon = 2^{-\Omega(n^{2\alpha})}$.[3] 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 1.4. *There exist constants $c > 0$ and $0 < \mu, \nu < 1$ such that for any $k \ge \log^c n$, there is an explicit deterministic (k,ϵ)-bit-fixing source extractor $E : \{0,1\}^n \to \{0,1\}^m$ where $m = k - O(k^\nu)$ and $\epsilon = O(k^{-\mu})$.*

We remark that using the technique of [13] 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 4.1.

1.5. 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".

1.5.1. 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 1.5 (seeded extractors). *Let $\mathcal{C}$ be a class of distributions on $\{0,1\}^n$. A function $E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a seeded ϵ-extractor for $\mathcal{C}$ if for every source*

3 It seems that a more careful analysis on our part would have given error $\epsilon = 2^{-\Omega(n^{2\gamma})}$.

$X \in \mathcal{C}$ the distribution $E(X,Y)$ (obtained by sampling x from X and a uniform $y \in_R \{0,1\}^d$ and computing $E(x,y)$) is ϵ-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 a lot of min-entropy.

Definition 1.6 (seeded extractors for high min-entropy sources). *The min-entropy of a distribution X over $\{0,1\}^n$ is $H_\infty(X) = min_{x \in \{0,1\}^n} \log(1/\Pr(x))$. A function $E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ϵ)-extractor if it is a seeded ϵ-extractor for the class of all sources X with $H_\infty(X) \geq k$.*

There are explicit constructions of seeded (k,ϵ)-extractors that use seed of length $\text{polylog}(n/\epsilon)$ to extract k random bits. The reader is referred to [24] 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 1.7 (seeded extractors for bit-fixing sources). *A function $E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a seeded (k,ϵ)-bit-fixing source extractor if it is a seeded ϵ-extractor for the class of all (n,k)-bit-fixing sources.*

1.5.2. 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 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 can be *very 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 which 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 uniformly distributed string.
- X' and Y are independent.

The precise definition is given in Definition 3.1.[4] Note that a seed obtainer reduces the task of constructing *deterministic* extractors into that of constructing *seeded* extractors. (In fact, it is even sufficient to construct seeded extractors for bit-fixing sources.) 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'.

1.5.3. 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 construction for 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 someone "randomly samples" a short piece from X. In such a case it is expected that the short piece contains a small fraction of the good bits. Let X' be the string that remains after "removing" the short piece. The important observation is that X' is a bit-fixing source that is *independent* from the output Z. It turns out that the same phenomena 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. Surprisingly, we show how to use the bits extracted by the extractor E to perform the 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 $\{1, \ldots, n\}$. (We elaborate on such sampling techniques later on.) The first output 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 $\{1, \ldots, n\}$ 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 in-

4 The precise definition is slightly more technical. We only require that X' is a convex combination of (n,k')-bit-fixing sources. Furthermore, the pair (X', Y) are allowed to be ϵ-close to a pair with the aforementioned properties.

dices 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 3. We remark that the analysis requires that the extractor E has error ϵ which is smaller than $2^{-|W|}$ (where $|W|$ is the number of bits used by the sampling method).

1.5.4. A deterministic extractor for large k (i.e. $k \gg \sqrt{n}$) Our first construction builds on the deterministic extractor of Kamp and Zuckerman [13] 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. We run the seed obtainer F on the input source to generate a bit-fixing source X' and a seed Y. We then extract all the randomness in X' by running a seeded extractor on X' using Y as seed.

1.5.5. 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 which 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 [13] can be applied when $k < \sqrt{n}$ but only give deterministic bit-fixing source extractors that extract very few bits ($\Omega(\log k)$ bits).[5]

Deterministic extractors that extract $\Omega(\log k)$ bits. Kamp and Zuckerman [13] consider the distribution obtained by using a bit-fixing source $X = (X_1, \cdots, X_n)$ 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, one uses X_i 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 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, using their technique with a short cycle of length $k^{1/4}$ produces an extractor construction which extracts $\log(k^{1/4}) = \Omega(\log k)$ bits.[6]

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 $\Omega(\log k)$ bits 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 [18, 19] the class of high min-entropy sources does not have even 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 deterministic extractor extracts these many bits as long as $k > \log^c n$ for some constant c.

The seeded extractor uses its seed to randomly partition the indices $\{1, \ldots, n\}$ into r sets $T_1, \cdots, T_r$ (for r equals, say, $\log^4 n$), 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 $\{1, \ldots, n\}$ into $r = \operatorname{poly}\log n$ sets $T_1, \ldots, T_r$ such that for any set $S \subseteq \{1, \ldots, n\}$ of size k, with high probability (probability at least $1 - 1/\log n$) all sets $T_1, \ldots, T_r$ contain approximately k/r indices from S.

Suppose we could afford using many random bits. A natural solution is to choose n random variables $V_1, \ldots, V_n \in \{1, \ldots, r\}$ 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 a union bound shows that with high probability all sets $T_1, \ldots, T_r$ 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 ϵ-close to being pairwise independent (for $\epsilon = 1/\log^a n$ for some sufficiently large constant a). Such variables can be constructed using only $O(\log\log n)$ [15, 1, 12] random bits and suffice to guarantee the required properties.

The same technique also gives us a method for sampling a set T of indices in $\{1, \ldots, n\}$. (Recall that we require such a sampling technique 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 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. [21] as a component in a construction of seeded extractors.

5 This was observed independently by Lipton and Vishnoi [14].

6 In fact, a similar idea is used in [13] in order to reduce the case of large d to the case of $d = 2$.

1.6. Outline

In Section 2 we define the notations used in this paper. In Section 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 4 we observe that the technique of [13] can be used to extract few bits even when k is small. In section 5 we give constructions for averaging samplers. In section 6 we give a construction of a seeded bit-fixing source extractor which makes use of the sampling techniques of Section 5. In Section 7 we plug all the components together and prove our main theorems. Finally, in section 8 we give some open problems. Due to space limitations some proofs are deferred to the full version.

2. preliminaries

Notations: We use $[n]$ to denote the set $\{1, \ldots, n\}$. 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 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.

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

$$\Pr_w(k_{min} \leq |Samp(w) \cap S| \leq k_{max}) \geq 1 - \delta$$

The definition above is nonstandard in several respects. The sampler is only required to work against sets of size k. This also allows k_{min} and k_{max} to depend on k. Furthermore, we implicitly require that the sampler has "distinct samples" and do not allow T to be a multi-set.[7]

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, w.h.p., the number of good bits in the subset is between k_{min} and k_{max}.

7 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 $k > \sqrt{n}$. Note that a random set of size roughly n/k will hit S with good probability.

2.2. Notation for Probability Distributions

Some of the proofs in this paper require some manipulations of probability distributions. We use the following notation. We denote the probability of an event B under a probability distribution P by $\Pr_P[B]$. A random variable R is a function $R : \Omega \to U$ (where Ω 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 probability distribution of R is equal to P.

Definition 2.2 (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 $R : \Omega \to U$. Given an event $A \subseteq \Omega$ we define $(R|A)$ to be the probability distribution over U given by $\Pr_{(R|A)}[B] = \Pr[R \in B|A]$.

We also need the notion of convex combination of distributions.

Definition 2.3 (convex combination of distributions). *Given distributions $P_1, \ldots, P_t$ over U and coefficients $\alpha_1, \ldots, \alpha_t \geq 0$ such that $\sum_{1 \leq i \leq t} \alpha_i = 1$ we 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 whose proof are omitted.

Lemma 2.4. *Let (A, B) be a random variable over $\{0,1\}^u \times \{0,1\}^v$ 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 ϵ-close to P. Then (A, B) is ϵ-close to (A, P) (here P is independent of A.)*

Lemma 2.5. *Let (A, B) be a random variable over $\{0,1\}^u \times \{0,1\}^v$ which is ϵ-close to (A', U_v) (where U_v is the uniform distribution on $\{0,1\}^v$ and is independent of A'). Then, for every $b \in \{0,1\}^v$ the distribution $(A|B = b)$ is $(\epsilon \cdot 2^{v+1})$-close to A'.*

3. Obtaining an independent seed

3.1. Seed obtainers

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 can be *very 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"* which 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 3.1 (seed obtainer). *A function $F : \{0,1\}^n \rightarrow \{0,1\}^n \times \{0,1\}^d$ is a (k,k',ρ)-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 η and α_a are nonnegative and $\eta + \sum_a \alpha_a = 1$) such that $\eta \leq \rho$ and for every a the distribution R_a is ρ-close to a distribution (U_a, V_a) where:*

- *U_a is an (n,k')-bit fixing source.*
- *V_a is uniformly distributed in $\{0,1\}^d$.*
- *U_a, V_a are independent.*

It follows that given a seed obtainer one can use a *seeded extractor* for bit fixing sources to construct a *deterministic* (or seedless) extractor for bit fixing sources. This is stated precisely in the next lemma whose proof is omitted.

Theorem 3.2. *Let $F : \{0,1\}^n \rightarrow \{0,1\}^n \times \{0,1\}^d$ be a (k,k',ρ)-seed obtainer. Let $E_1 : \{0,1\}^n \times \{0,1\}^d \rightarrow \{0,1\}^m$ be a (k',ϵ) seeded 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*

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 bit-fixing source extractor. We now show how to convert any deterministic bit-fixing source extractor with sufficiently small error into a seed obtainer. The motivation is that a deterministic bit-fixing source extractor which extracts *very few* bits gives rise to a seed obtainer which in turn can be used to construct a deterministic bit-fixing source extractor which extracts *almost all* the randomness from a given source.

Our construction appears in Figure 1. In words, given x, the seed obtainer first computes $z = E(x)$. It uses a part of z as the output y and another part to sample a substring of x. It obtains x' by erasing the sampled substring from x. The basic idea is that the extractor E only "needs" k_{min} bits out of the k random bits in the source. By the properties of the sampler, when given a random input w, with high probability T_w contains k_{min} random bits. We can therefore hope to argue that the extractor "extracts" the bits from T_w and thus, the output y is independent of $x' = x_{[n]\setminus T_w}$ which contains $k - k_{max}$ bits.

Ingredients:

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

Result: A (k, k', ρ)-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 $z = E(x)$. Let $d = m - t$. We identify $z \in \{0,1\}^m$ with $(w, y) \in \{0,1\}^t \times \{0,1\}^d$.
- Let $T_w = Samp(w)$.
- Let $x' = x_{[n]\setminus T_w}$. If $|x'| < n$ we pad it with zeroes to get an n-bit long string.
- Output x', y.

Figure 1. A seed obtainer for (n,k)-bit-fixing sources

Theorem 3.3 (construction of seed obtainers). *For every $k < n$, Let $Samp$ and E be as in Figure 1 (that is, $Samp : \{0,1\}^t \rightarrow P([n])$ is an $(n, k, k_{min}, k_{max}, \delta)$-sampler and $E : \{0,1\}^n \rightarrow \{0,1\}^m$ is a (k_{min}, ϵ)-bit-fixing source extractor). Then, $F : \{0,1\}^n \rightarrow \{0,1\}^n \times \{0,1\}^d$ defined in Figure 1 is a (k, k', ρ)-seed-obtainer for $d = m - t$, $k' = k - k_{max}$ and $\rho = \max(\epsilon + \delta, \epsilon \cdot 2^{t+1})$.*

3.3. Proof of Theorem 3.3

In this section we prove Theorem 3.3. Let E be a bit-fixing source extractor and $Samp$ be a sampler which satisfy the requirements in Theorem 3.3. 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. More precisely, we set $Z = E(X)$ and split Z into two parts $(W, Y) \in \{0,1\}^t \times \{0,1\}^d$. 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 $x_{T'_a}$. Let $X' = X'_W$. Our goal is to show that the pair (X', Y) is close to a convex combination of pairs

(U, V) where U is a bit fixing source, V is uniform, and U and V are independent.

Definition 3.4. *We say that a string $a \in \{0,1\}^t$ correctly splits X if $k_{min} \le |S \cap T_a| \le k_{max}$.*

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 Z 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 3.5. *For every fixed $a \in \{0,1\}^t$ which correctly splits X the pair of random variables (X'_a, Z) is ϵ-close to the pair (X'_a, U_m) where U_m is uniformly distributed over $\{0,1\}^m$ and is independent of X'_a.*

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 σ in the indices of T_a and σ' in the indices of T'_a. More formally, we denote the ℓ indices of T_a by $i_1 < i_2 < \cdots < i_\ell$ and the $n-\ell$ indices of T'_a by $i'_1 < i'_2 < \cdots < i'_{n-\ell}$. 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 $(X'_a, Z) = (X'_a, E(X)) = (X'_a, E([X_a; X'_a]))$. 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

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

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 $(Z|X'_a = b)$ is ϵ-close to uniform. We now apply Lemma 2.4 with $A = X'_a$ and $B = Z$ and conclude that the pair (X'_a, Z) is ϵ-close to (X'_a, U_m). □

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

Lemma 3.6. *For every fixed $a \in \{0,1\}^t$ that correctly splits X, the distribution $((X'_a, Y)|W = a)$ is $\epsilon 2^{t+1}$-close to (X'_a, U_d) (where U_d is uniformly distributed over $\{0,1\}^d$ and independent of X'_a).*

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 $\{W = a\}$ occurs with non-zero probability as W is ϵ-close to uniform over $\{0,1\}^t$. The Lemma will follow as a straightforward application of Lemma 2.5. We set $A = (X'_a, Y)$, $B = W$ and $A' = (X'_a, U_d)$. We indeed have that (A, B) is ϵ-close to (A', U_t) and the lemma follows. □

We are now ready to prove Theorem 3.3.

Proof. (of Theorem 3.3) By the properties of the extractor we have that W is ϵ-close to uniform. It follows (by the properties of the sampler) that the probability that W correctly splits X is $1 - \eta$ for some η which satisfies $\eta \le \epsilon + \delta$. We now consider the output random variable $R = (X', Y)$. 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 $(R|$“W doesn’t correctly split X”). For every correctly splitting a we set R_a to be the distribution $(R|W = a)$ and $\alpha_a = \Pr[W = a]$. 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 $R_a = ((X', Y)|W = a) = ((X'_W, Y)|W = a) = ((X'_a, Y)|W = a)$. By Lemma 3.6 we have that R_a is $\epsilon \cdot 2^{t+1}$-close to (X'_a, U_d). 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. □

4. Extracting few bits for any k

The deterministic bit-fixing source extractor of Kamp and Zuckerman [13] only works for $k > \sqrt{n}$. However, the technique of [13] easily gives a deterministic bit-fixing source extractor which 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 which will enable us to extract many more bits.[8]

Theorem 4.1. *For every $n > k \ge 100$ there is an explicit deterministic $(k, 2^{-\sqrt{k}})$-bit-fixing source extractor $E : \{0,1\}^n \to \{0,1\}^{(\log k)/4}$.*

For the proof, we need the following Lemma.

Lemma 4.2. *[13][9] Let the graph G be an odd cycle with M vertices and second eigenvalue λ. Suppose we take a walk on G for n steps, starting from some fixed vertex v with the*

8 We have learned that this was observed independently by Lipton and Vishnoi [14].

9 This Lemma is a special case of Lemma 3.3 in [13] with $\epsilon = 0$, $d = 2$ and fixed initial distributions.

steps taken according to the symbols from 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 few bits from a bit-fixing source X, we will use the bits of X to conduct a random walk on a small cycle.[10]

Proof. (of Theroem 4.1) 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 $\cos(\frac{\pi}{d})$. Using the previous lemma, 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 $0 < x < 1$

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

Therefore

$$\left(\cos\left(\frac{\pi}{\sqrt[4]{k}}\right)\right)^k \leq \left(1 - \frac{\pi^2}{4\sqrt{k}}\right)^k$$
$$\leq \left(e^{-\frac{\pi^2}{4\sqrt{k}}}\right)^k = \left(e^{-\frac{\pi^2}{4}}\right)^{\sqrt{k}}$$
$$\leq 4^{-\sqrt{k}}$$

where the second to last inequality holds because $(1 - x) \leq e^{-x}$ 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. □

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 2.1).

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

We give two constructions of such schemes. The first is based on ℓ-wise independence that spends t random bits (for any $t > \log n$) and achieves error $\delta = 2^{-\Omega(t/\log n)}$. Our construction of deterministic bit-fixing source extractors for small k requires schemes which use only $\alpha \log k$ bits for some small constant $\alpha > 0$. Our second construction works by derandomizing our first construction using "almost 2-wise dependent variables" [15, 1, 12]. We now explain our sampling technique and note that a similar idea was used in [21].

Bellare and Rompel [3] gave a sampler construction based on ℓ-wise independent variables. We use a twist on their method: Suppose we are aiming to hit k/r bits when given a subset S of size k. We generate ℓ-wise independent variables $Z_1, \cdots, Z_n \in [r]$ and define $T = \{j|Z_j = 1\}$. It follows that with high probability $S \cap T$ is of size approximately k/r. This is stated formally below. Due to space limitations the proofs are omitted.[11]

Lemma 5.1 (Sampling with small δ). *For every integers n, k, r, t such that $r \leq k \leq n$ and $4 \log n \leq t \leq \frac{k \log n}{20r}$ there is an explicit $(n, k, \frac{1}{2} \cdot \frac{k}{r}, 3 \cdot \frac{k}{r}, 2^{-\Omega(t/\log n)})$-sampler which uses t random bits.*

We use the intuition explained above to derandomize this construction. This gives schemes that use only $O(\log k)$ bits.

Lemma 5.2 (sampling with $O(\log k)$ bits). *Fix any constant $0 < \alpha < 1$. There exist constants $c > 0$ and $0 < b, e < 1$(depending on α) such that for any $k \geq \log^c n$, we obtain an $(n, k, k^e/2, 3 \cdot k^e, O(k^{-b}))$-sampler $Samp : \{0,1\}^t \to P([n])$ where $t = \alpha \cdot \log k$ and e satisfies $e > 1/2$.*

Lemma 5.3 (partitioning with $O(\log k)$ bits). *Fix any constant $0 < \alpha < 1$. There exist constants $c > 0$ and $0 < b, e < 1$(depending on α) such that for any $k \geq \log^c n$, we*

10 In fact, this idea was already used in [13] to convert bit-fixing sources into "approximate symbol fixing sources".

11 We remark that this construction is different than the common way of using ℓ-wise independence for sampling [3]. The more common way is to take n/r random variables $V_1, \ldots, V_{n/r} \in [n]$ which are ℓ-wise independent and sample the set $T = \{V_1, \ldots, V_{n/r}\}$. The expected size $|S \cap T|$ is k/r and one gets the same probability of success $\delta = 2^{-\Omega(\ell)}$ by the tail inequality of [3]. The two methods require roughly the same number of random bits. Nevertheless, the method of Lemma 5.1 has several advantages:

- The method used in Lemma 5.1 can also be used for partitioning: one simply sets $T_i = \{j|Z_j = i\}$.
- The method used in Lemma 5.1 guarantees that T is a set whereas the second method may produce a multi-set.
- The method used in Lemma 5.1 can be derandomized and use much fewer bits (at least for small r and large δ). More precisely, suppose that $r \leq \log n$ and say $\ell = 2$. In this range of parameters, one can use $O(\log \log n)$ random bits to generate n variables $Z_1, \cdots, Z_n \in \{0,1\}^{\log r}$ which are $(1/\log n)$-close to being pairwise independent. Thus, the same technique can be used to construct more randomness efficient samplers (at the cost of having larger error parameter δ.) We remark that in the case of the standard method no savings can be made as it requires variables Z_i over $\{0,1\}^{\log n}$ and even sampling one such variable requires $\log n$ random bits.

can use $\alpha\cdot\log k$ random bits to partition $[n]$ into $m = \Omega(k^b)$ sets $T_1, \ldots, T_m$ such that for any $S \subseteq [n]$ where $|S| = k$

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

where e satisfies $e > 1/2$.

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.

Lemma 6.1. *Fix any constant $0 < \alpha < 1$. There exist constants $c > 0$ and $0 < b < 1$ (depending on α) such that for any $k \geq \log^c n$, there exists a seeded (k, ϵ)-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 $\epsilon = O(k^{-b})$.*

Let X be an (n, k)-bit-fixing source. Let $x = x_1, \ldots, x_n$ be a string sampled by X. E works as follows: We use lemma 5.3 to construct a partition of the bits of x. Then we output the xor of 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 5.3 when using the lemma with the parameter α.

E(x,y)

- Obtain a partition of $[n]$ into $m = \Omega(k^b)$ sets $T_1, \ldots, T_m$ using lemma 5.3 with the parameter α.
- For $1 \leq i \leq m$, compute $z_i = \oplus_{j \in T_i} x_j$.
- Output $z = z_1, \ldots, z_m$.

The proof of Lemma 6.1 is omitted.

7. Putting it all Together

We now explain how to use the ingredients to prove our two main Theorems. Exact details appear in the full version. To prove Theorem 1.3 we start with the deterministic bit-fixing source extractor of [13]. We use the sampler of Lemma 5.1 to transform it into a seed obtainer, and then extract many bits by applying the seeded extractor of [20].

To prove Theorem 1.4 we start with the deterministic bit-fixing source extractor of Theorem 4.1. We use the sampler of Lemma 5.2 to transform it into a seed obtainer, and then extract more bits by applying the seeded extractor of Lemma 6.1. We then convert the deterministic bit-fixing source extractor we just generated into a seed obtainer and use the seeded extractor of [20] to extract many bits.

8. Discussion and open problems

We give explicit constructions of deterministic bit-fixing source extractors which extract almost all the randomness. However, we achieve rather large error $\epsilon = k^{-\Omega(1)}$ for $k < \sqrt{n}$. In general, when starting from an initial deterministic bit-fixing source extractor which extracts m_0 bits, using our construction of seed obtainers can only give $\rho \geq 2^{-m_0}$ (recall that ρ measures the statistical distance of the seed obtainer from the desired distribution. This in turn places a similar bound on the error of the final extractor. Loosely speaking, the cause of this limitation is that a sampler which uses m_0 random bits cannot have error parameter $\delta < 2^{-m_0}$. Thus, a natural way to reduce the error in our construction for small k is to come up with an initial deterministic extractor that extracts more bits which will enable the seed obtainer construction to use samplers with smaller error parameter δ.

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 even in our first construction. Nevertheless, we point out that the focus of this paper is extracting as many bits as possible and when $m = k - u$ even the nonexplicit result achieves error $\epsilon \geq 2^{-u}$ and thus cannot achieve $\epsilon < 2^{-m}$ when $m \geq k/2$. We do not know whether this is the case for every deterministic bit-fixing source extractor construction. We remark that the nonexplicit result is not optimal for $m = 1$ as exhibited by the xor-extractor which is errorless (see also [8]). We find it interesting to achieve $m = \Omega(k)$ with $\epsilon = 2^{-\Omega(k)}$.

Another interesting open problem is whether our technique of seed obtainers is applicable for other types of sources. For example, a natural way to improve the output length of "standard" seeded extractors is to first run an extractor with a short seed to extract few bits and then use these bits as a seed to a second extractor (that uses a large seed) and extracts all the bits. It seems that our technique is not applicable in this case because it requires the first extractor to have very small error. More precisely, our analysis requires that the first extractor has error $\epsilon < 2^{-m}$ when extracting m bits. Nevertheless, by the lower bounds of [18, 19] no such seeded extractors exist.

Acknowledgements

The third author is grateful to David Zuckerman for very helpful discussions. We are grateful to Oded Goldreich and to anonymous referees for helpful comments.

References

[1] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k-wise independent random variables. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, volume II, pages 544–553, 1990.

[2] B. Barak, R. Impagliazzo, and A. Wigderson. Extracting randomness from few independent sources. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, 2004.

[3] M. Bellare and J. Rompel. Randomness-efficient oblivious sampling. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, 1994.

[4] M. Blum. Independent unbiased coin flips from a correlated biased source: a finite state Markov chain. In *Proceedings of the 25th Annual IEEE Symposium on Foundations of Computer Science*, pages 425–433, 1984.

[5] V. Boyko. On the security properties of OAEP as an all-or-nothing transform. In *Proc. 19th International Advances in Cryptology Conference – CRYPTO '99*, pages 503–518, 1999.

[6] R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai. Exposure-resilient functions and all-or-nothing transforms. *Lecture Notes in Computer Science*, 1807, 2000.

[7] B. Chor and O. Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, Apr. 1988. Special issue on cryptography.

[8] B. Chor, O. Goldreich, J. Hastad, J. Friedman, S. Rudich, and R. Smolensky. The bit extraction problem or t-resilient functions. In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science*, 1985.

[9] A. Cohen and A. Wigderson. Dispersers, deterministic amplification, and weak random sources. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, 1989.

[10] Y. Dodis. *Exposure-Resilient Cryptography*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, Aug. 2000.

[11] Y. Dodis, A. Sahai, and A. Smith. On perfect and adaptive security in exposure-resilient cryptography. *Lecture Notes in Computer Science*, 2045, 2001.

[12] S. Even, O. Goldreich, M. Luby, N. Nisan, and B. Velickovic. Efficient approximation of product distributions. *RSA: Random Structures & Algorithms*, 13, 1998.

[13] J. Kamp and D. Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, 2003.

[14] R. Lipton and N. Vishnoi. Manuscript. 2004.

[15] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 213–223, 1990.

[16] N. Nisan. Extracting randomness: How and why: A survey. In *Proceedings of the 11th Annual IEEE Conference on Computational Complexity*, pages 44–58, 1996.

[17] N. Nisan and A. Ta-Shma. Extracting randomness: A survey and new constructions. *Journal of Computer and System Sciences*, 58, 1999.

[18] N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.

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

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

[21] O. Reingold, R. Shaltiel, and A. Wigderson. Extracting randomness via repeated condensing. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, 2000.

[22] R. Rivest. All-or-nothing encryption and the package transform. In *Fast software encryption: 4th International Workshop, FSE*, volume 1267 of *Lecture Notes in Computer Science*, 1997.

[23] M. Santha and U. V. Vazirani. Generating quasi-random sequences from semi-random sources. *Journal of Computer and System Sciences*, 33:75–87, 1986.

[24] R. Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the EATCS*, 77:67–95, 2002.

[25] L. Trevisan and S. Vadhan. Extracting randomness from samplable distributions. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, 2000.

[26] S. Vadhan. Randomness extractors and their many guises. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 9–12, 2002.

[27] U. Vazirani. Efficient considerations in using semi-random sources. In *Proceedings of the 19th Annual ACM Symposium on the Theory of Computing*, 1987.

[28] U. Vazirani. Strong communication complexity or generating quasi-random sequences from two communicating semi-random sources. *Combinatorica*, 7:375–392, 1987.

[29] U. Vazirani and V. Vazirani. Random polynomial time is equal to semi-random polynomial time. Technical Report TR88-959, Cornell University, Computer Science Department, Dec. 1988.

[30] J. von Neumann. Various techniques used in connection with random digits. *Applied Math Series*, 12:36–38, 1951.

[31] D. Zuckerman. General weak random sources. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 534–543, 1990.

[32] D. Zuckerman. Simulating BPP using a general weak random source. *Algorithmica*, 16(4/5):367–391, Oct./Nov. 1996.