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

Ariel Gabizon[†] Ran Raz[‡] Ronen Shaltiel[§]

August 18, 2005

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 [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 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 $k \le \sqrt{n}$ 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.

[*]A preliminary version of this paper appeared in FOCS 2004.

[†]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 $A \subseteq \{0,1\}^m$, $|P(A)-Q(A)| \le \epsilon$.

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 [37] who showed how to use many independent tosses of a biassed coin (with unknown bias) to obtain an unbiased coin. Blum [6] considered sources that are generated by a finite Markov-chain. Santha and Vazirani [29], Vazirani [34, 35], Chor and Goldreich [10], Barak et al. [2], Barak et al. [3] and Raz [25] studied sources that are composed of several independent samples from various classes of distributions. Trevisan and Vadhan [31] 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 [36, 10, 12, 38, 39] and was explicitly defined by Nisan and Zuckerman [23].) 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 [21, 22, 30, 32] for various surveys on randomness extractors.

1.3 Bit-fixing sources

In this paper we concentrate on the family of "bit-fixing sources" introduced by Chor et al. [11]. A distribution X over $\{0,1\}^n$ is 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 extractors). *A distribution X over $\{0,1\}^n$ is an (n,k)-* bit-fixing source *if there exists a subset $S = \{i_1,\ldots,i_k\} \subseteq \{1,\ldots,n\}$ such that $X_{i_1}, X_{i_2}, \ldots, X_{i_k}$ 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 [11]. 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 [28, 7, 8, 14]. The reader is referred to [13] 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,ϵ)-bit-fixing source extractor with $\epsilon < 1/2$ 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. [11] 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.

[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 (cf. [5]). In this paper we are only concerned with the "oblivious case".

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, 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 [17] 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 $0 < \alpha < 1/2$ the extractor only extracts $m = \eta n^{2\alpha}$ bits.

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 $0 < \gamma < 1/2$ there exists an integer n' (depending on γ) such that for any $n > n'$ and 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 $0 < \alpha < 1/2$. We can choose any $\gamma < \alpha$ and extract $m = n^{1/2+\alpha} - n^{1/2+\gamma}$ bits whereas the construction of [17] only extracts $m = O(n^{2\alpha})$ bits. For this choice of parameters we achieve error $\epsilon = 2^{-\Omega(n^\gamma)}$ whereas [17] 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 1.4. *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, 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 [17] 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 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 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 high min-entropy.

Definition 1.6 (seeded extractors for high min-entropy sources). *The min-entropy of a distribution X over $\{0,1\}^n$ is $H_\infty(X) = min_{x \in \{0,1\}^n} \log_2(1/\Pr(x))$. 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 (k,ϵ)-extractors that use seed of length polylog(n/ϵ) to extract k random bits. The reader is referred to [30] 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 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 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 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 string that is almost uniformly distributed.
- X' and Y are almost independent.

The precise definition is slightly more technical and is given in Definition 3.1. Note that a seed obtainer reduces the task of constructing *deterministic* extractors into 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.)

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 transformation when applied to a specific deterministic bit-fixing source extractor. Consider the "xor-extractor" $E(x) = \oplus_{1 \le i \le n} x_i$. 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 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 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 $\{1, \ldots, n\}$. 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 $\{1, \ldots, n\}$ 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 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 3. We remark that the analysis requires that the extractor E has error ϵ that 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 bit-fixing source extractor of Kamp and Zuckerman [17] 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.

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 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 [17] 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)[3].

Deterministic extractors that extract $\Omega(\log k)$ bits. Kamp and Zuckerman [17] consider the distribution obtained by using a bit-fixing source $X = (X_1, \ldots, X_n)$ 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 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.[4]

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 [23, 24] 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 $\{1, \ldots, n\}$ into r sets $T_1, \ldots, T_r$ (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 $r = \mathrm{poly}\log n$ 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 $1 - O(1/\log n)$) 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 by a union bound one can show 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 $O(\log\log n)$ random bits [20, 1, 15] and suffice to guarantee the required properties.

[3]This was observed independently by Lipton and Vishnoi [18].

[4]In fact, a similar idea is used in [17] in order to reduce the case of large d to the case of $d = 2$.

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 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. [27] as a component in a construction of seeded extractors.

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 [17] 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 that 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.

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 paper 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 Ω and O signs only to denote absolute constants (i.e., not depending 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.

2.1 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 $|S| = k$:*

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

The definition above is nonstandard in several respects. In the more common definition (c.f. [16] a sampler is required to work for sets of arbitrary size. In the definition above (which is similar in spirit to the one in [33]) 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 has "distinct samples" as we do not allow T to be a multi-set.[5]

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}.

2.2 Probability distributions

Some of the proofs in this paper 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 $R : \Omega \to U$ (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 distribution of the output of R is equal to P. Given two random variables R_1, R_2 over the same probability space Ω 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 Ω_1, Ω_2 we define $P_1 \otimes P_2$ to be the product distribution of P_1 and P_1 which is defined over the domain $\Omega_1 \times \Omega_2$.

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 $\Pr_{(R|A)}[B] = \Pr_R[R \in B|A]$.

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.

Lemma 2.4. *Let X, Y and V be distributions over $\{0, 1\}^n$ such that X is ϵ-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.

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

□

[5]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 good probability.

Lemma 2.5. *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 ϵ-close to P. Then (A, B) is ϵ-close to $(A \otimes P)$.*

Proof.

$$\frac{1}{2} \cdot \sum_{a,b} |\Pr[(A, B) = (a, b)] - \Pr_{A \otimes P}[a, b]| = \frac{1}{2} \cdot \sum_{a,b} |\Pr[A = a]\Pr[B = b|A = a] - \Pr[A = a]\Pr_P[b]|$$

$$\le \frac{1}{2} \cdot \sum_a \Pr[A = a] \sum_b |\Pr[B = b|A = a] - \Pr_P[b]| \le \epsilon/2$$

□

Lemma 2.6. *Let (A, B) be a random variable that takes values in $\{0,1\}^u \times \{0,1\}^v$ which is ϵ-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 α-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 ϵ-close to U_v and therefore $\Pr_B[B = b^*] \ge 2^{-v} - \epsilon$. Thus,

$$\Pr_{(A,B)}[D'] - \Pr_{(A',U_v)}[D'] = \Pr_{(A|B=b^*)}[D] \Pr_B[B = b^*] - \Pr_{A'}[D] \cdot 2^{-v}$$

$$\ge \Pr_{(A|B=b^*)}[D](2^{-v} - \epsilon) - \Pr_{A'}[D] \cdot 2^{-v} \ge 2^{-v}[\Pr_{(A|B=b^*)}[D] - \Pr_{A'}[D]] - \epsilon$$

By our assumption the expression in square brackets is at least α and thus,

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

Thus, we get a contradiction. □

3 Obtaining an independent seed

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 3.1 (seed obtainer). *A function $F : \{0,1\}^n \to \{0,1\}^n \times \{0,1\}^d$ 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 $\eta \le \rho$ and for every a there exists an (n, k')-bit-fixing source Z_a such that R_a is ρ-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 3.2. *Let $F : \{0,1\}^n \to \{0,1\}^n \times \{0,1\}^d$ be a (k, k', ρ)-seed obtainer. Let $E_1 : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be a seeded (k', ϵ)-bit-fixing source extractor. Then $E : \{0,1\}^n \to \{0,1\}^m$ 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 $\eta \le \rho$. 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 2.4 we have that $E(X)$ is $(2\eta + \epsilon + \rho)$-close to uniform. The lemma follows because $2\eta + \epsilon + \rho \le \epsilon + 3\rho$. □

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

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

Ingredients:

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

Result: A (k,k',ρ)-seed obtainer $F : \{0,1\}^n \to \{0,1\}^n \times \{0,1\}^{m-t}$ 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.

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. 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 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 $E(X)$ 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 3.5. *For every fixed $a \in \{0,1\}^t$ which correctly splits X the pair of random variables $(X'_a, E(X))$ is ϵ-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 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 $(X'_a, E(X)) = (X'_a, E([X_a; X'_a]))$. 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 ϵ-close to uniform. We now apply Lemma 2.5 with $A = X'_a$ and $B = E(X)$ and conclude that the pair $(X'_a, E(X))$ is ϵ-close to $(X'_a \otimes U_m)$. □

We now argue that if ϵ 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 3.6. *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 non-zero probability as $E_1(X)$ is ϵ-close to uniform over $\{0,1\}^t$. The lemma will follow as a straightforward application of Lemma 2.6. 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 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 $E_1(X)$ is ϵ-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 η which satisfies $\eta \le \epsilon + \delta$. 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|"$E_1(X)$ 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 3.6 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. □

4 Extracting few bits for any k

The deterministic bit-fixing source extractor of Kamp and Zuckerman [17] 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 4.1. *For every $n > k \geq 100$ 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 result, which is a very special case of Lemma 3.3 in [17].

Lemma 4.2. *([17, Lemma 3.3] for $\epsilon = 0$ and $d = 2$.) 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 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 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 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 $\cos(\frac{\pi}{d})$ ([19, Ex. 11.1]). Using Lemma 4.2, 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. $\square$

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 $[k_{min}, k_{max}]$. 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 5.1 (sampling with low error). *Fix any constants $0 < \gamma \leq 1/2$ and $\alpha > 0$. There exists a constant n' depending on α and γ, 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 $Samp : \{0,1\}^t \to P([n])$ where $t = \alpha \cdot n^{2\gamma}$.*

The second constructions 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 5.1 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 5.2 (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 α) 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 $Samp : \{0,1\}^t \to P([n])$ where $t = \alpha \cdot \log k$.*

Lemma 5.3 (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 α) 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 $T_1, \ldots, T_m$ such that for any $S \subseteq [n]$ where $|S| = k$*

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

The first construction is based on "ℓ-wise independence", and the second is based on "almost 2-wise dependence" [20, 1, 15]. Sampling techniques based on ℓ-wise independence were first suggested by Bellare and Rompel [4]. However, this technique is not good enough in our setting and we use a different approach (which was also used in [27] with slightly different parameters). In appendix A we explain the approach in detail, compare it to the approach of [4] and give full proofs of the Lemmas above.

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 6.1. *Fix any constant $0 < \alpha < 1$. There exist constants $c > 0$ and $0 < b < 1$ (both depending on α) such that for any $n \geq 16$ and $k \geq \log^c n$, there exists an explicit seeded (k, ϵ)-bit-fixing source extractor $E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ 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 $x = x_1, \ldots, x_n$ 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 5.3 when using the lemma with the parameter α.

E(x,y)

- We use the seed y to 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 $z = z_1, \ldots, z_m$.

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 5.3, when $k \geq \log^c n$, $\Pr(A) \geq 1 - O(k^{-b})$. Therefore, by Lemma 2.4

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

□

7 Deterministic extractors for bit-fixing sources

In this section, we compose the ingredients from previous sections to prove Theorems 1.3 and 1.4. Namely, given choices for a deterministic bit-fixing source extractor, sampler and seeded bit-fixing source extractor, we use Theorems 3.2 and 3.3 to get a new deterministic bit-fixing source extractor. This works as follows: We "plug in" a deterministic extractor that extracts little randomness and sampler into Theorem 3.3 to get a seed obtainer. We then "plug in" this seed obtainer and a seeded extractor into Theorem 3.2 to get a new deterministic extractor which extracts almost all of the randomness. It is convenient to express this composition as follows:

Theorem 7.1. *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}, ϵ^*)-bit-fixing source extractor $E^* : \{0,1\}^n \rightarrow \{0,1\}^{m'}$.*
- *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, ϵ)-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 3.3 to get a $(k, k - k_{max}, \max(\epsilon^* + \delta, \epsilon^* \cdot 2^{t+1}))$-seed obtainer $F : \{0,1\}^n \to \{0,1\}^n \times \{0,1\}^{m'-t}$. Since $m' - t \geq d$, we can use F and E_1 in Theorem 3.2 to obtain a deterministic (k, ϵ)-bit-fixing source extractor $E : \{0,1\}^n \to \{0,1\}^m$ where $\epsilon = \epsilon_1 + 3 \cdot \max(\epsilon^* + \delta, \epsilon^* \cdot 2^{t+1})$. □

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

Theorem 7.2. *[26] For any n, k and $\epsilon > 0$, there exists a (k, ϵ)-extractor $Ext : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ where $m = k$ and $d = O(\log^2 n \cdot \log(1/\epsilon) \cdot \log k)$*

7.1 An extractor for large k (proof of Theorem 1.3)

To prove Theorem 1.3, we first state results about the required ingredients and then use the ingredients in Theorem 7.1.

We use the deterministic bit-fixing source extractor of Kamp and Zuckerman [17]. Loosely speaking, the following theorem states that when $k >> \sqrt{n}$, we can deterministically extract a polynomial fraction of the randomness with an exponentially small error.

Theorem 7.3. *[17] 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, ϵ^*)-bit-fixing source extractor $E^* : \{0,1\}^n \to \{0,1\}^m$ 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 ℓ-wise independence from Lemma 5.1. We use the seeded extractor of [26] (Theorem 7.2) which we now restate in the following form:

Corollary 7.4. *Fix any constants $0 < \gamma \leq 1/2$ and $\alpha > 0$. There exists a constant n' depending on γ such that for any integers n, k satisfying $n > n'$ and $k \leq n$ there exists a (k, ϵ_1)-extractor $E_1 : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ 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 7.2. 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 γ), we can take $\epsilon_1 = 2^{-\frac{\alpha \cdot n^\gamma}{c_1}} = 2^{-\Omega(\alpha \cdot n^\gamma)}$. □

We now compose the ingredients from Theorem 7.3, Lemma 5.1 and Corollary 7.4 to prove Theorem 1.3. 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 1.3) Let c be the constant in Theorem 7.3. We use Theorem 7.1 with the following ingredients:

- The $(n, k, (n^{1/2+\gamma})/6, n^{1/2+\gamma}, \delta = 2^{-\Omega(n^\gamma)})$-sampler $Samp : \{0,1\}^t \to P([n])$ from Lemma 5.1 where $t = (c/72)n^{2\gamma}$.

- The deterministic $((n^{1/2+\gamma})/6, \epsilon^* = 2^{-m'})$-bit-fixing source extractor $E^* : \{0,1\}^n \to \{0,1\}^{m'}$ from Theorem 7.3 where $m' = (c/36)n^{2\gamma}$.

- The $(k - n^{1/2+\gamma}, \epsilon_1 = 2^{-\Omega(n^\gamma)})$-extractor $E_1 : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ from Corollary 7.4 with $d \le (c/72)n^{2\gamma}$ and $m = k - n^{1/2+\gamma}$.

Note that all three objects exist for a large enough n depending only on γ (c is a universal constant). Note that $m' \ge t+d$. Therefore, applying Theorem 7.1, we get a deterministic (k, ϵ)-bit-fixing source extractor $E : \{0,1\}^n \to \{0,1\}^m$ where $m = k - n^{1/2+\gamma}$ and

$$\begin{aligned}\epsilon = \epsilon_1 + 3 \cdot \max(\epsilon^* + \delta, \epsilon^* \cdot 2^{t+1}) &= 2^{-\Omega(n^\gamma)} + 3 \cdot \max\left(2^{-(c/36)n^{2\gamma}} + 2^{-\Omega(n^\gamma)}, 2^{-(c/36)n^{2\gamma}} \cdot 2^{(c/72)n^{2\gamma}+1}\right) \\ &= 2^{-\Omega(n^\gamma)} + 3 \cdot \max\left(2^{-\Omega(n^\gamma)}, 2^{-(c/72)n^{2\gamma}+1}\right) \\ &= 2^{-\Omega(n^\gamma)}\end{aligned}$$

(for a large enough n depending on γ). □

7.2 An extractor for small k (proof of Theorem 1.4)

To prove Theorem 1.4 we need a deterministic bit-fixing source extractor for $k < \sqrt{n}$. We use the extractor of Theorem 4.1. We prove the Theorem in two steps. First, we use Theorem 7.1 to convert the initial extractor into a deterministic bit-fixing source extractor that extracts more bits. We then apply Theorem 7.1 again to obtain a deterministic bit-fixing source extractor which 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 $k \ge \log^c n$ for some constant c.

Lemma 7.5. *There exist constants $c, b > 0$ such that for any $k \ge \log^c n$ and large enough n, there exists an explicit deterministic (k, k^{-b})-bit-fixing source extractor $E : \{0,1\}^n \to \{0,1\}^m$ 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 4.1. We will need $c_1 \cdot \log\log n$ random bits to use the sampler of Lemma 5.2 and the seeded extractor of Theorem 6.1 (for some constant c_1). Thus, when $k \ge \log^c n$ for large enough c, we will have enough bits.

Formally, we use Theorem 7.1 with the following ingredients:

- The $(n, k, k^e/2, 3 \cdot k^e, \delta = k^{-\Omega(1)})$-sampler $Samp : \{0,1\}^t \to P([n])$ from Lemma 5.2 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 $E^* : \{0,1\}^n \to \{0,1\}^{m'}$ from Theorem 4.1 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 $E_1 : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ from Theorem 6.1 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 7.1 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 5.2). Applying Theorem 7.1, we get a deterministic (k, ϵ)-bit-fixing source extractor $E : \{0,1\}^n \to \{0,1\}^m$. 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. □

We now compose the ingredients from Lemmas 5.2 and 7.5, and Theorem 7.2 to prove Theorem 1.4. 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 7.5. We want $\log^5 n$ random bits to use the seeded extractor of Theorem 7.2. Thus, when $k \geq \log^c n$ for large enough c, we will have enough bits.

Proof. (of Theorem 1.4) Let b be the constant in Lemma 7.5. We use Theorem 7.1 with the following ingredients:

- The $(n, k, k^e/2, 3 \cdot k^e, \delta = k^{-\Omega(1)})$-sampler $Samp : \{0,1\}^t \to P([n])$ from Lemma 5.2 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 $E^* : \{0,1\}^n \to \{0,1\}^{m'}$ from Lemma 7.5 where $m' = (k^e/2)^{\Omega(1)}$.
- The $(k - 3 \cdot k^e, \epsilon_1 = 1/n)$-extractor $E_1 : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ from Theorem 7.2 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 7.1 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 7.1, we get a deterministic (k, ϵ)-bit-fixing source extractor $E : \{0,1\}^n \to \{0,1\}^m$, where

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

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

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 that $k < \sqrt{n}$. 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 δ 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 can the error 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 which is errorless (see also [11]). Given the discussion above we find it interesting to achieve $m = \Omega(k)$ with $\epsilon = 2^{-\Omega(k)}$ for every choice of k.

Acknowledgements

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

References

[1] N. Alon, O. Goldreich, J. Hastad, and R. Peralta. Simple constructions of almost k−wise independent random variables. *Journal of Random Structures and Algorithms*, 3(3):289–304, 1992.

[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*, pages 384–393, 2004.

[3] B. Barak, G. Kindler, R. Shaltiel, B. Sudakov, and A. Wigderson. Simulating independence: New constructions of condenesers, ramsey graphs, dispersers, and extractors. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 1–10, 2005.

[4] M. Bellare and J. Rompel. Randomness-efficient oblivious sampling. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 276–287, 1994.

[5] M. Ben-Or and N. Linial. Collective coin flipping. *ADVCR: Advances in Computing Research*, 5:91–115, 1989.

[6] 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.

[7] 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.

[8] R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai. Exposure-resilient functions and all-or-nothing transforms. *Lecture Notes in Computer Science*, 1807:453–469, 2000.

[9] I. L. Carter and M. N. Wegman. Universal classes of hash functions. In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing*, pages 106–112, 1977.

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

[11] 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*, pages 396–407, 1985.

[12] 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*, pages 14–25, 1989.

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

[14] Y. Dodis, A. Sahai, and A. Smith. On perfect and adaptive security in exposure-resilient cryptography. *Lecture Notes in Computer Science*, 2045:299–322, 2001.

[15] S. Even, O. Goldreich, M. Luby, N. Nisan, and B. Velickovic. Efficient approximation of product distributions. *Random Structures & Algorithms*, 13:1–16, 1998.

[16] O. Goldreich. A sample of samplers - a computational perspective on sampling. *Electronic Colloquium on Computational Complexity (ECCC)*, 4(20), 1997.

[17] 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*, pages 92–101, 2003.

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

[19] L. Lovasz. *Combinatorial Problems and Exercises*. North-Holland, Amsterdam, 1979.

[20] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, August 1993.

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

[22] N. Nisan and A. Ta-Shma. Extracting randomness: A survey and new constructions. *Journal of Computer and System Sciences*, 58(1):148–173, 1999.

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

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

[25] R. Raz. Extractors with weak random seeds. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 11–20, 2005.

[26] 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.

[27] 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*, pages 149–158, 2000.

[28] 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*, pages 210–218, 1997.

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

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

[31] L. Trevisan and S. Vadhan. Extracting randomness from samplable distributions. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 32–42, 2000.

[32] 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.

[33] S. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *J. Cryptology*, 17(1):43–77, 2004.

[34] U. Vazirani. Efficient considerations in using semi-random sources. In *Proceedings of the 19th Annual ACM Symposium on the Theory of Computing*, pages 160–168, 1987.

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

[36] U. V. Vazirani and V. V. Vazirani. Random polynomial time is equal to semi-random polynomial time. In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science*, pages 417–428, 1985.

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

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

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

A Sampling and partitioning

In this section we give constructions of samplers and prove Lemmas 5.1,5.2 and 5.3.

A.1 Sampling using ℓ-wise independence

Bellare and Rompel [4] 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 $Z_1, \ldots, Z_n \in [r]$ and define $T = \{i | Z_i = 1\}$. It follows that with high probability $S \cap T$ is of size approximately k/r. This is stated formally in the following Lemma. (We explain the difference between this method and that of [4] in Remark A.3.)

Lemma A.1. *For every integers n, k, r, t such that $r \leq k \leq n$ and $6 \log n \leq t \leq \frac{k \log n}{20r}$ 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 a seed of t random bits.*

Before proving this Lemma we show that Lemma 5.1 is a special case.

Proof. (of Lemma 5.1) We use Lemma A.1 with the parameters n, k and $r = \frac{3k}{n^{1/2+\gamma}}$, $t = \alpha \cdot n^{2\gamma}$. We need to check that $6 \log n \leq t \leq \frac{k \log n}{20r}$. Clearly, $t \geq 6 \log n$ (for a large enough n depending on α and γ). On the other hand,

$$\frac{k \log n}{20r} = \frac{n^{1/2+\gamma} \log n}{60} \geq \alpha \cdot n^{2\gamma} = t$$

(for a large enough n depending on α and γ). Thus, applying lemma A.1, we get an $(n, k, k/2r, 3k/r, \delta)$-sampler $Samp : \{0,1\}^t \to P([n])$ where

$$\delta = 2^{-\Omega(t/\log n)} = 2^{-\Omega(\alpha \cdot n^{2\gamma}/\log n)} = 2^{-\Omega(\alpha \cdot n^{\gamma})}$$

(for a large enough n depending on α and γ). $\square$

We need the following tail-inequality for ℓ-wise independent variables due to Bellare and Rompel [4].

Theorem A.2. *[4] Let $\ell \geq 6$ be an even integer. Suppose that $X_1, \ldots, X_n$ are ℓ-wise independent random variables taking values in $[0,1]$. Let $X = \sum_{1 \leq i \leq n} X_i$, and $\mu = E(X)$, and let $A > 0$. Then*

$$\Pr[|X - \mu| \geq A] \leq 8 \left(\frac{\ell\mu + \ell^2}{A^2} \right)^{\ell/2}$$

We now prove Lemma A.1.

Proof. (of Lemma A.1) Let ℓ be the largest even integer such that $\ell \log n \leq t$ and let $q = \lfloor \log r \rfloor \leq \log n$. There are constructions which use $\ell \log n \leq t$ random bits to generate n random variables $Z_1 \ldots, Z_n \in \{0,1\}^q$ that are ℓ-wise independent [9]. The sampler generates such random variables. Let $a \in \{0,1\}^q$ be some fixed value. We define a random variable $T = \{i | Z_i = a\}$. Let $S \subseteq [n]$ be some subset of size k. For $1 \leq i \leq n$ we define a boolean random variable X_i such that $X_i = 1$ if $Z_i = a$. Let $X = |S \cap T| = \sum_{i \in S} X_i$. Note that $\mu = E(X) = k/2^q$ and that the random variables $X_1, \ldots, X_n$ are ℓ-wise independent. Applying Theorem A.2 with $A = k/2r$ we get that

$$\Pr[|X - \mu| \geq A] \leq 8 \left(\frac{\ell k/2^q + \ell^2}{A^2} \right)^{\ell/2}$$

Note that

$$\{|X-\mu|<A\} \subseteq \left\{\frac{k}{2^q}-A<X<\frac{k}{2^q}+A\right\} \subseteq \left\{\frac{k}{r}-A<X<\frac{2k}{r}+A\right\} \subseteq \{k_{min} \leq X \leq k_{max}\}$$

for $k_{min}=k/2r$ and $k_{max}=3k/r$. Note that $\ell \leq \frac{t}{\log n} \leq \frac{k}{20r}$. We conclude that

$$\Pr[k_{min} \leq |S\cap T| \leq k_{max}] \geq 1-8\left(\frac{\ell\frac{k}{2^q}+\ell^2}{(\frac{k}{2r})^2}\right)^{\ell/2} \geq 1-8\left(\frac{4r^2(\frac{2\ell k}{r}+\frac{\ell k}{20r})}{k^2}\right)^{\ell/2}$$

$$\geq 1-8\left(\frac{10\ell r}{k}\right)^{\ell/2} \geq 1-2^{-(\ell/2+3)} \geq 1-2^{-\Omega(t/\log n)}$$

$\square$

Remark A.3. *We remark that this construction is different than the common way of using ℓ-wise independence for sampling [4]. 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 multi-set $T=\{V_1,\ldots,V_{n/r}\}$. The expected size of the multi-set $|S\cap T|$ is k/r and one gets the same probability of success $\delta = 2^{-\Omega(\ell)}$ by the tail inequality of [4]. The two methods require roughly the same number of random bits. Nevertheless, the method of Lemma A.1 has the following advantages:*

- *It can also be used for partitioning.*
- *The method used in Lemma A.1 guarantees that T is a set whereas the standard method may produce a multi-set.*
- *The method used in Lemma A.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 $Z_1,\ldots,Z_n \in \{0,1\}^{\log r}$ 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 use this idea in Section A.2. 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.*

A.2 Sampling and partitioning using fewer bits

We now derandomize the construction of Lemma A.1 to give schemes which use only $O(\log k)$ bits and prove Lemmas 5.2 and 5.3. These two Lemmata follow from the following more general Lemma.

Lemma A.4. *Fix any integer $n \geq 16$. Let k be an integer such that $k \leq n$. Let r satisfy $r \leq k$. Let r' be the power of 2 that satisfies $(1/2)r < r' \leq r$. Let $\epsilon > 0$ satisfy $1/kr \leq \epsilon \leq 1/8r$. We can use $7\log r + 3(\log\log n + \log(1/\epsilon))$ random bits to explicitly partition $[n]$ into r' sets $T_1,\ldots,T_{r'}$ such that for any $S \subseteq [n]$ where $|S| = k$*

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

We prove Lemma A.4 in the next section. We now explain how the two Lemmata follow from Lemma A.4.

Proof. (of Lemma 5.3) Set $b = \alpha/38$. Use Lemma A.4 with the parameters $r = k^b$ and $\epsilon = k^{-4b}$ to obtain a partition $T_1, \ldots, T_{r'}$ of $[n]$ where $(1/2)r < r' \leq r$ is a power of 2.

To use Lemma A.4 with these parameters we need $7\log r + 3(\log\log n + \log(1/\epsilon)) = 7\log k^b + 3(\log\log n + \log k^{4b})$ random bits.
We want to use at most $\alpha \cdot \log k$ bits.
Set $c = 6/\alpha$. Since we assume that $k \geq \log^c n$

$$(\alpha/2)\log k \geq (\alpha/2)(6/\alpha)\log\log n = 3\log\log n$$

So now we need

$$(\alpha/2)\log k \geq 7\log k^b + 3\log k^{4b} = b(7+12)\log k$$

Or, equivalently

$$b \leq \alpha/38$$

Set $e = 1 - b$. So $k/2r = k^e/2$ and $3k/r = 3 \cdot k^e$. Note that $e > 1/2$ as required.
Using Lemma A.4

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

□

Lemma 5.2 easily follows from Lemma 5.3.

Proof. (of Lemma 5.2) Use Lemma 5.3 with the parameters n, k and α to obtain a partition of $[n]$ $T_1, \ldots, T_m$ and take T_1 as the sample. It is immediate that the required parameters are achieved. □

Proof of Lemma A.4

The sampler construction in Lemma A.1 relied on random variables $Z_1, \ldots, Z_n \in [r]$ which are ℓ-wise independent. We now show that we can derandomize this construction and get a (weaker) sampler by using $Z_1, \ldots, Z_n$ which are only *pair-wise ϵ-dependent*. Naor and Naor [20] (and later Alon et al. [1]) gave constructions of such variables using very few random bits. This allows us to reduce the number of random bits required for sampling and partitioning.

The following definition formalizes a notion of limited independence, slightly more general than the one discussed above:

Definition A.5. ***(ℓ-wise ϵ-dependent variables)***. *Let D be a distribution. We say that the random variables $Z_1, \ldots, Z_n$ are ℓ-wise ϵ-dependent according to D if for every $M \subseteq [n]$ such that $|M| \leq \ell$ the distribution Z_M (that is, the joint distribution of the Z_i's such that $i \in M$) is ϵ-close to the distribution $D^{\otimes |M|}$, i.e., the distribution of $|M|$ independent random variables chosen according to D. We sometimes omit D when it is the uniform distribution.*
Random bit variables $B_1, \ldots, B_n$ are ℓ-wise ϵ-dependent with mean p, if they are ℓ-wise ϵ-dependent according to the distribution $D = (1-p, p)$ on $\{0, 1\}$.

We need two properties about ℓ-wise ϵ-dependent variables: That they can be generated using very few random bits and that their sum is concentrated around the expectation. The first property is proven in Lemma A.7 and the second in Lemma A.8.

The following theorem states that ℓ-wise ϵ-dependent bit variables can be generated by very few random bits.

Theorem A.6. *([1])* [6]

For any $n \geq 16$, $\ell \geq 1$ and $0 < \epsilon < 1/2$, ℓ-wise ϵ-dependent bits $B_1, \ldots, B_n$ can be generated using $3(\ell + \log\log n + \log(1/\epsilon))$ truly random bits.

We can generate pair-wise ϵ-dependent variables in larger domains using ℓ-wise ϵ-dependent bit variables.[7]

Lemma A.7. *Let $r < n$ be a power of 2. For any $n \geq 16$ and $0 < \epsilon < 1/2$, we can generate pair-wise ϵ-dependent variables $Z_1, \ldots, Z_n \in [r]$ using $7\log r + 3(\log\log n + \log(1/\epsilon))$ truly random bits.*

Proof. Using Theorem A.6, we generate $2\log r$-wise ϵ-dependent bit variables $B_1, \ldots, B_{n\log r}$ using $3(2\log r + \log\log(n\log r) + \log(1/\epsilon)) \leq 7\log r + 3(\log\log n + \log(1/\epsilon))$ bits. We partition the B_i's into n blocks of size $\log r$ and interpret the i'th block as a value $Z_i \in [r]$.
The joint distribution of the bits in any block or 2 blocks is ϵ-close to uniform. Therefore, the Z_i's are pair-wise ϵ-dependent.

□

In the following lemma, we use Chebichev's inequality to show that the sum of pair-wise ϵ-dependent bit variables is close to its expectation with high probability.

Lemma A.8. *Let p satisfy $0 < p < 1$. Let $\epsilon > 0$ satisfy $p/k \leq \epsilon \leq p/4$. Let $B_1, \ldots, B_k$ be pair-wise ϵ-dependent bit variables with mean p. Let $B = \sum_{i=1}^{k} B_i$.*
Then

$$\Pr(|B - pk| > pk/2) = O(\epsilon/p^2).$$

Proof. Using linearity of expectation we get $|E(B) - pk| \leq \epsilon k$.
Therefore

$$\Pr(|B - pk| > pk/2) \leq \Pr(|B - E(B)| > pk/2 - \epsilon k)$$

So it's enough to bound

$$\Pr(|B - E(B)| > pk/2 - \epsilon k)$$

Fix any $i, j \in [k]$ where $i \neq j$. The covariance of B_i and B_j will be small since they are almost independent:

$$cov(B_i, B_j) = E(B_i \cdot B_j) - E(B_i)E(B_j)$$

[6] The theorem is stated a bit differently and only for odd ℓ in ([1]) but this form is easily deduced from theorem 3 in that paper observing that $(\ell + 1)$-wise ϵ-dependence implies ℓ-wise ϵ-dependence.

[7] Actually, a construction of such (and more general types of) variables already appears in [15].

$$= \Pr(B_i = 1; B_j = 1) - \Pr(B_i = 1)\Pr(B_j = 1)$$
$$\leq (p^2 + \epsilon) - (p - \epsilon)^2 = (1 + 2p - \epsilon)\epsilon \leq 3\epsilon$$

(where the second equality is because B_i and B_j are bit variables)

Therefore, the variance of B won't be too large:

$$Var(B) = \sum_i Var(B_i) + \sum_{i \neq j} cov(B_i, B_j) \leq (p + \epsilon)k + 3\epsilon k^2 \leq pk + 4\epsilon k^2$$

Therefore, by Chebichev's inequality

$$\Pr(|B - E(B)| > pk/2 - \epsilon k) < \frac{pk + 4\epsilon k^2}{(pk/2 - \epsilon k)^2}$$

We required that $\epsilon \leq p/4$ and therefore

$$\leq \frac{pk + 4\epsilon k^2}{(pk/4)^2} = O(1/pk) + O(\epsilon/p^2) = O(\epsilon/p^2)$$

(where the last equality follows by the requirement that $\epsilon \geq p/k$) □

Now we can easily prove Lemma A.4.

Proof. (of Lemma A.4) Let r' be the power of 2 in the statement of the lemma. Using Lemma A.7, we generate pair-wise ϵ-dependent $Z_1, \ldots, Z_n \in [r']$. For $1 \leq i \leq r'$, we define $T_i = \{j | Z_j = i\}$.
Assume, w.l.o.g., that $S = \{1, \ldots, k\}$.
Given $i \in [r']$, define the bit variables $B_1, \ldots, B_k$ by $B_j = 1 \Leftrightarrow Z_j = i$.
It is easy to see that the B_j's are pairwise 2ϵ-dependent with mean $1/r'$.
Define $C_i = \sum_{j=1}^{k} B_j$. Note that $C_i = |T_i \cap S|$.
Notice that $1/r'$ and 2ϵ satisfy the requirements in Lemma A.8.
Using Lemma A.8

$$\Pr(|C_i - k/r'| > k/2r') = O(\epsilon \cdot (r')^2) = O(\epsilon \cdot r^2)$$

Using the union bound

$$\Pr(\exists i \ s.t \ |C_i - k/r'| > k/2r') = O(\epsilon \cdot r^3)$$

Thus, we can obtain a partition $T_1, \ldots, T_{r'}$ of $[n]$ such that, with probability at least $1 - O(\epsilon \cdot r^3)$

$$\forall i \ k/2r' \leq |T_i \cap S| \leq 3k/2r'$$

Which implies that with at least the same probability

$$\forall i \ k/2r \leq |T_i \cap S| \leq 3k/r$$

□