Electronic Colloquium on Computational Complexity, Report No. 176 (2024)

Nearly-Linear Time Seeded Extractors with Short Seeds

Dean Doron* João Ribeiro†

Abstract

Seeded extractors are fundamental objects in pseudorandomness and cryptography, and a deep line of work has designed polynomial-time seeded extractors with nearly-optimal parameters. However, existing constructions of seeded extractors with short seed length and large output length run in time $\Omega(n \log(1/\varepsilon))$ and often slower, where n is the input source length and ε is the error of the extractor. Since cryptographic applications of extractors require ε to be small, the resulting runtime makes these extractors unusable in practice.

Motivated by this, we explore constructions of strong seeded extractors with short seeds computable in nearly-linear time $O(n \log^c n)$, for any error ε. We show that an appropriate combination of modern condensers and classical approaches for constructing seeded extractors for high min-entropy sources yields strong extractors for n-bit sources with any min-entropy k and any target error ε with seed length $d = O(\log(n/\varepsilon))$ and output length $m = (1-\eta)k$ for an arbitrarily small constant $\eta > 0$, running in nearly-linear time, after a reasonable one-time preprocessing step (finding a primitive element of $\mathbb{F}_q$ with $q = \mathrm{poly}(n/\varepsilon)$ a power of 2) that is only required when $k < 2^{C \log^* n} \cdot \log^2(n/\varepsilon)$, for a constant $C > 0$ and $\log^*$ the iterated logarithm, and which can be implemented in time $\mathrm{polylog}(n/\varepsilon)$ under mild conditions on q. As a second contribution, we give an instantiation of Trevisan's extractor that can be evaluated in *truly* linear time in the RAM model, as long as the number of output bits is at most $\frac{n}{\log(1/\varepsilon)\,\mathrm{polylog}(n)}$. Previous fast implementations of Trevisan's extractor ran in $\widetilde{O}(n)$ time in this setting. In particular, these extractors directly yield privacy amplification protocols with the same time complexity and output length, and communication complexity equal to their seed length.

*Ben-Gurion University. deand@bgu.ac.il. Part of this work was done while visiting Instituto de Telecomunicações and the Simons Institute for the Theory of Computing.

†Instituto de Telecomunicações and Departamento de Matemática, Instituto Superior Técnico, Universidade de Lisboa. jribeiro@tecnico.ulisboa.pt. Part of this work was done while at NOVA LINCS and NOVA School of Science and Technology, and while visiting the Simons Institute for the Theory of Computing.

ISSN 1433-8092

Contents

1 Introduction **3**
- 1.1 Our Contributions . . . 4
- 1.2 Other Related Work . . . 5
- 1.3 Technical Overview . . . 6
- 1.4 Future Work . . . 8
- 1.5 Acknowledgements . . . 8

2 Preliminaries **8**
- 2.1 Notation . . . 8
- 2.2 Model of Computation . . . 8
- 2.3 Fast Finite Fields Operations . . . 9
- 2.4 Statistical Distance, Entropy . . . 9
- 2.5 Extractors and Condensers . . . 10
- 2.6 Averaging Samplers . . . 11
- 2.7 Standard Composition Techniques for Extractors . . . 13

3 Additional Building Blocks **14**
- 3.1 Fast Generation of Small-Bias Sets . . . 14
- 3.2 A Sampler from Bounded Independence . . . 15
- 3.3 Nearly-Linear Time Condensers . . . 18

4 A Faster Instantiation of Trevisan's Extractor **19**

5 Nearly-Linear Time Extractors with Order-Optimal Seed Length **20**
- 5.1 A Non-Recursive Construction . . . 20
  - 5.1.1 Item 2: Generating the block source . . . 21
  - 5.1.2 Item 3: Subsampling from the block source . . . 23
  - 5.1.3 Item 4: Applying a block source extractor . . . 24
  - 5.1.4 Improving the output length . . . 25
- 5.2 A Recursive Construction . . . 26
  - 5.2.1 The (extremely) low-error case . . . 26
  - 5.2.2 The (relatively) high-error case . . . 29

1 Introduction

Seeded randomness extractors are central objects in the theory of pseudorandomness. A strong (k, ε)-seeded extractor is a deterministic function $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ that receives as input an n-bit source of randomness X with k bits of min-entropy[1] and a d-bit independent and uniformly random seed Y, and outputs an m-bit string $\mathsf{Ext}(X, Y)$ that is ε-close in statistical distance to the uniform distribution over $\{0,1\}^m$, where ε is an error term, even when the seed Y is revealed. Besides their most direct application to the generation of nearly-perfect randomness from imperfect physical sources of randomness (and their inaugural applications to derandomizing space-bounded computation [NZ96] and privacy amplification [BBCM95]), seeded extractors have also found many other surprising applications throughout computer science, particularly in cryptography.

For most applications, it is important to minimize the *seed length* of the extractor. A standard application of the probabilistic method shows the existence of strong (k, ε)-seeded extractors with seed length $d = \log(n-k) + 2\log(1/\varepsilon) + O(1)$ and output length $m = k - 2\log(1/\varepsilon) - O(1)$, and we also know that these parameters are optimal up to the $O(1)$ terms [RT00]. This motivated a deep line of research devising explicit constructions of seeded extractors with seed length as small as possible spanning more than a decade (e.g., [NZ96, SZ99, NT99, Tre01, TZS06, SU05]) and culminating in extractors with essentially optimal seed length [LRVW03, GUV09]. In particular, the beautiful work of Guruswami, Umans, and Vadhan [GUV09] gives explicit strong extractors with order-optimal seed length $d = O(\log(n/\varepsilon))$ and output length $m = (1-\eta)k$ for any constant $\delta > 0$, and follow-up work [DKSS13, TU12] further improved the *entropy loss* $k + d - m$. The extractors constructed in these works are explicit, in the sense that there is an algorithm that given x and y computes the corresponding output $\mathsf{Ext}(x, y)$ in time polynomial in the input length.

A closer look shows that the short-seed constructions presented in the literature all run in time $\Omega(n\log(1/\varepsilon))$, and often significantly slower. In cryptographic applications of extractors we want the error guarantee ε to be small, which means that implementations running in time $\Omega(n\log(1/\varepsilon))$ are often impractical. If we insist on nearly-linear runtime for arbitrary error ε, we can use strong seeded extractors based on universal hash functions that can be implemented in $O(n\log n)$ time (e.g., see [HT16]), have essentially optimal output length, but have the severe drawback of requiring a very large seed length $d = \Omega(m)$.

These limitations have been noted in a series of works studying concrete implementations of seeded extractors, with practical applications in quantum cryptography in mind [MPS12, FWE+23, FYEC24]. For example, Foreman, Yeung, Edgington, and Curchod [FYEC24] implement a version of Trevisan's extractor [Tre01, RRV02] with its standard instantiation of Reed–Solomon codes concatenated with the Hadmadard code, and emphasize its excessive running time as a major reason towards non-adoption.[2] Instead, they have to rely on extractors based on universal hash functions, which, as mentioned above, are fast but require very large seeds.

This state of affairs motivates the following question, which is the main focus of this work:

> *Can we construct strong (k, ε)-seeded extractors with seed length $d = O(\log(n/\varepsilon))$ and output length $m = (1-\eta)k$ computable in nearly-linear time, for arbitrary error ε?*

Progress on this problem would immediately lead to faster implementations of many cryptographic protocols that use seeded extractors.

[1] A random variable X has k bits of min-entropy if $\Pr[X = x] \le 2^{-k}$ for all x. Min-entropy has been the most common measure for the quality of a weak source of randomness since the work of Chor and Goldreich [CG88].

[2] The reason why these works focus on Trevisan's extractor is that this is the best seeded extractor (in terms of asymptotic seed length) that is known to be secure against quantum adversaries [DPVR12].

1.1 Our Contributions

We make progress on the construction of nearly-linear time extractors.

Seeded extractors with order-optimal seed length and large output length. We construct nearly-linear time strong seeded extractors with order-optimal seed length and large output length for any k and ε, with the caveat that they require a one-time preprocessing step whenever $k = O(\log^2(n/\varepsilon))$. This preprocessing step corresponds to finding primitive elements of finite fields $\mathbb{F}_q$ with $q = \mathrm{poly}(n/\varepsilon)$, which, as we discuss below, is reasonable in practical applications. More precisely, we have the following result.

Theorem 1. *For any constant $\eta > 0$ there exists a constant $C > 0$ such that the following holds. For any positive integers n and $k \le n$ and any $\varepsilon > 0$ satisfying $k \ge C \log(n/\varepsilon)$ there exists a strong (k, ε)-seeded extractor*

$$\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$$

with seed length $d \le C \log(n/\varepsilon)$ and output length $m \ge (1-\eta)k$. Furthermore,

- *if $k \ge 2^{C \log^* n} \cdot \log^2(n/\varepsilon)$, then Ext is computable in time $\widetilde{O}(n)$, where $\widetilde{O}(\cdot)$ hides polylogarithmic factors in its argument and $\log^*$ denotes the iterated logarithm;*
- *if $k < 2^{C \log^* n} \cdot \log^2(n/\varepsilon)$, then Ext is computable in time $\widetilde{O}(n)$ after a preprocessing step, corresponding to finding a primitive element of $\mathbb{F}_q$ with $q = \mathrm{poly}(n/\varepsilon)$ a power of 2.*[3]

Theorem 1 follows from combining modern condensers with short seeds (namely, the lossless condenser of Kalev and Ta-Shma [KT22] and the lossy Reed-Solomon-based condenser of Guruswami, Umans, and Vadhan [GUV09]) with a careful combination and instantiation of classical recursive approaches developed by Srinivasan and Zuckerman [SZ99] and in [GUV09]. It readily implies, among other things, an $\widetilde{O}(n)$-time privacy amplification protocol where only $O(\log(n/\varepsilon))$ bits need to be communicated over the one-way authenticated public channel and almost all the min-entropy can be extracted (after a reasonable one-time preprocessing step if the min-entropy bound k is very small).

A new non-recursive construction. As a conceptual contribution which may be of independent interest, we present a new "non-recursive" construction of extractors with seed length $O(\log(n/\varepsilon))$ and output length $(1-\eta)k$ that is computable in nearly-linear time when $k > \mathrm{polylog}(1/\varepsilon)$ and avoids the complicated recursive procedures from [SZ99, GUV09]. We believe this to be a conceptually better approach towards constructing seeded extractors, and we discuss it in more detail in the technical overview.

[3] In full rigor, the preprocessing step corresponds to finding primitive elements of $O(\log\log n)$ fields $\mathbb{F}_q$ with orders $q \le \mathrm{poly}(n/\varepsilon)$, each a power of 2. This $O(\log\log n)$ term has negligible influence on the complexity of this preprocessing step. Note that we can find such a primitive element in time $\mathrm{polylog}(n/\varepsilon)$ if $q \le \mathrm{poly}(n/\varepsilon)$ is a power of 2 and we know the factorization of $q-1$, but we do not know how to do that in time $\widetilde{O}(\log q)$. More precisely, given the factorization of $q-1$ we can test whether a given $\alpha \in \mathbb{F}_q$ is primitive in time $\mathrm{polylog}(q)$ by checking whether $\alpha^{\frac{q-1}{p}} \ne 1$ for all prime factors p of $q-1$. We can exploit this in various ways. If we are fine with using randomness in the one-time preprocessing stage, then we can sample an element of $\mathbb{F}_q$ uniformly at random, test whether it is primitive, and repeat if not. If we insist on a deterministic algorithm, then we can combine the testing procedure with algorithms of Shoup [Sho90] or Shparlinski [Shp92] which identify in time $\mathrm{polylog}(q)$ a subset of size $\mathrm{polylog}(q)$ in $\mathbb{F}_q$ that is guaranteed to contain a primitive element. For an alternative faster randomized algorithm, see [DD06].

Faster instantiations of Trevisan's extractor. One of the most widely-used explicit seeded extractors is Trevisan's extractor [Tre01, RRV02]. While by now we have extractors with better parameters, one of its main advantages is that it is one of the few examples of extractors, and in a sense the best one, which are known to be *quantum proof*.[4]

Trevisan's extractor uses two basic primitives: combinatorial designs (when more than one output bit is desired), and binary list-decodable codes. A standard instantiation of such suitable codes goes by concatenating a Reed-Solomon code with a Hadamard code, and this is also what is considered in [FWE+23, FYEC24]. As they also observe, this gives a nearly-linear time construction when the output length $m = 1$. In fact, by leveraging fast multipoint evaluation, one can also get a nearly-linear time construction for any output length $m \leq \frac{n}{\log(1/\varepsilon)}$, although this was not noted in previous works.[5]

Our main contribution in this direction is an alternative instantiation of Trevisan's extractor that can be computed in truly linear time on a RAM in the logarithmic cost model, for any output length $m \leq \frac{n}{\log(1/\varepsilon)\cdot \mathrm{polylog}(n)}$.

Theorem 2. *There exists an instantiation of Trevisan's extractor, set to extract m bits with any error $\varepsilon > 0$, that is computable in:*

1. *Time $O(n) + m\log(1/\varepsilon)\cdot\mathrm{polylog}(n)$ after a preprocessing step running in time $\widetilde{O}(m\log(n/\varepsilon))$, on a RAM in the logarithmic cost model. In particular, there exists a universal constant c, such that whenever $m \leq \frac{n}{\log(1/\varepsilon)\cdot\log^c(n)}$, the instantiation runs in time $O(n)$, without the need for a preprocessing step.*
2. *Time $\widetilde{O}(n + m\log(1/\varepsilon))$ in the Turing model.*

We note that one interesting instantiation of the above theorem is when Trevisan's extractor is set to output $k^{\Omega(1)}$ bits for $k = n^{\Omega(1)}$. In this setting, Trevisan's extractor requires a seed of length $O\left(\frac{\log^2(n/\varepsilon)}{\log(1/\varepsilon)}\right)$, and, as long as ε is not too tiny, we get truly-linear runtime.

1.2 Other Related Work

Besides the long line of work focusing on improved constructions of explicit seeded extractors and mentioned in the introduction above, other works have studied randomness extraction in a variety of restricted computational models. These include extractors computable by streaming algorithms [BRST02], local algorithms [Lu02, Vad04, BG13, CL18], AC^0 circuits [GVW15, CL18, CW24], AC^0 circuits with a layer of parity gates [HIV22], NC^1 circuits [CW24], and low-degree polynomials [ACG+22, AGMR24, GGH+24]. Moreover, implementations in various restricted computational models of other fundamental pseudorandomness primitives such as k-wise and ε-biased generators, that often play a key role in constructions of various types of extractors, have also been independently studied (see [HV06, Hea08, CRSW13, MRRR14] for a very partial list).

As mentioned briefly above, some works have also focused on constructing seeded extractors computable in time $O(n\log n)$ motivated by applications in privacy amplification for quantum key distribution. Such constructions are based on hash functions, and are thus far restricted to $\Omega(m)$ seed length. The work of Hayashi and Tsurumaru [HT16] presents an extensive discussion of such

[4] An extractor is quantum proof if its output is close to uniform even in the presence of a quantum adversary that has some (bounded) correlation with X. A bit more formally, Ext is quantum-proof if for all classical-quantum state ρ_{XE} (where E is a quantum state correlated with X) with $H_\infty(X|E) \geq k$, and a uniform seed Y, it holds that $\rho_{\mathsf{Ext}(X,Y)YE} \approx_\varepsilon \rho_{U_m}\otimes\rho_Y\otimes\rho_E$. See [DPVR12] for more details.

[5] For a rigorous statement on fast multipoint evaluation, see Lemma 2.1.

efforts. We also mention that nearly-linear time extractors with very short seed, in the regime $k = n^{\Omega(1)}$ and $\varepsilon = n^{-o(1)}$, were given in [DMOZ22], with applications in derandomization.

1.3 Technical Overview

In a nutshell, we obtain Theorem 1 by following two standard high-level steps:

1. We apply a randomness condenser with small seed length $O(\log(n/\varepsilon))$ to the original n-bit weak source X to obtain an output X' that is ε-close to a high min-entropy source.

2. We apply a seeded extractor tailored to high min-entropy sources with small seed length $O(\log(n/\varepsilon))$ to X' to obtain a long output that is ε-close to uniform.

To realize this approach, we need to implement each of these steps in nearly-linear time $\widetilde{O}(n)$ (possibly after a reasonable one-time preprocessing step). We briefly discuss how we achieve this, and some pitfalls we encounter along the way.

Observations about nearly-linear time condensers. In order to implement Item 1, we need to use *fast* condensers with short seeds. Luckily for us, some existing state-of-the-art constructions of condensers already satisfy this property, although, to the best of our knowledge, this has not been observed before. We argue this carefully in Section 3.3.

For example, the "lossy Reed-Solomon condenser" from [GUV09] interprets the source as a polynomial $f \in \mathbb{F}_q[x]$ of degree $d \leq n/\log q$ and the seed y as an element of $\mathbb{F}_q$, and outputs $\mathsf{RSCond}(f, y) = (f(y), f(\zeta y), \ldots, f(\zeta^{m'} y))$, for an appropriate m' and field size q, with ζ a primitive element of $\mathbb{F}_q$. Evaluating $\mathsf{RSCond}(f, y)$ corresponds to evaluating the same polynomial f on multiple points in $\mathbb{F}_q$. This is an instance of the classical problem of multipoint evaluation in computational algebra, for which we know fast and practical algorithms (e.g., see [vzGG13, Chapter 10] or Lemma 2.1) running in time $\widetilde{O}((d + m') \log q) = \widetilde{O}(n)$, since $d \leq n/\log q$ and if $m' \leq n/\log q$.

A downside of this condenser is that it requires knowing a primitive element ζ of $\mathbb{F}_q$ with $q = \mathrm{poly}(n/\varepsilon)$. As discussed above, if we know the factorization of $q - 1$ and q is a power of 2, then we can find such a primitive element in time $\mathrm{polylog}(q)$. Beyond that, having access to such primitive elements, which only need to be computed once independently of the source and seed, is reasonable in practice. Therefore, we may leave this as a one-time preprocessing step.

The lossless "KT condenser" from [KT22] has a similar flavor. It interprets the source as a polynomial $f \in \mathbb{F}_q[x]$ and the seed y as an evaluation point, and outputs $\mathsf{KTCond}(f, y) = (f(y), f'(y), \ldots, f^{(m')}(y))$, for some appropriate m'. The problem of evaluating several derivatives of the same polynomial f on the same point y (sometimes referred to as Hermite evaluation) is closely related to the multipoint evaluation problem above, and can also be solved in time $\widetilde{O}(n)$.[6] Evaluating the KT condenser does not require preprocessing. On the other hand, it only works when the min-entropy $k \geq C \log^2(n/\varepsilon)$ for a large constant $C > 0$, where n is the source length and ε the target error of the condenser.

The "ideal" approach to seeded extraction from high min-entropy sources. We have seen that there are fast condensers with short seeds. It remains to realize Item 2. Because of the initial condensing step, we may essentially assume that our n-bit weak source X has min-entropy $k \geq (1 - \delta)n$, for an arbitrarily small constant $\delta > 0$. In this case, we would like to realize in time

[6] Interestingly, recent works used other useful computational properties of the KT condenser. Cheng and Wu [CW24] crucially use the fact that the KT condenser can be computed in NC^1. Doron and Tell [DT23] use the fact that the KT condenser is logspace computable for applications in space-bounded derandomization.

$\widetilde{O}(n)$ and with overall seed length $O(\log(n/\varepsilon))$ what we see as the most natural approach to seeded extraction from high min-entropy sources:

1. Use a fresh short seed to transform X into a *block source* $Z = Z_1 \circ Z_2 \circ \cdots \circ Z_t$ with geometrically decreasing blocks, where $\circ$ denotes string concatenation. A block source has the property that each block Z_i has good min-entropy even conditioned on the values of blocks $Z_1, \ldots, Z_{i-1}$.

2. Perform *block source extraction* on Z using another fresh short seed. Due to its special structure, we can extract a long random string from Z using only the (small) seed length associated with extracting randomness from the smallest block Z_t, which has length $O(\log(n/\varepsilon))$.

The classical approach to Item 2 where we iteratively apply extractors based on universal hash functions with increasing output lengths to the blocks of Z from right to left is easily seen to run in time $\widetilde{O}(n)$ and requires a seed of length $O(\log(n/\varepsilon))$ if, e.g., we use the practical extractors of [TSSR11, HT16]. Therefore, we only need to worry about realizing Item 1.

A standard approach to Item 1 would be to use an *averaging sampler* to iteratively sample subsequences of X as the successive blocks of the block source Z, following a classical strategy of Nisan and Zuckerman [NZ96] (improved by [RSW06, Vad04]). We do know averaging samplers running in time $\widetilde{O}(n)$ (such as those based on random walks on a carefully chosen expander graph). However, this approach requires a fresh seed of length $\Theta(\log(n/\varepsilon))$ *per block of* Z. Since Z will have roughly $\log n$ blocks, this leads to an overall seed of length $\Theta(\log^2 n + \log(1/\varepsilon))$, which is too much for us.

Instead, we provide a new analysis of a sampler based on bounded independence, that runs in time $\widetilde{O}(n)$ and only requires a seed of length $O(\log(n/\varepsilon))$ to create the *entire* desired block source. We give the construction, which may be of independent interest, in Section 3.2. The caveat of this "block source creator" is that it only works as desired when the target error $\varepsilon \geq 2^{-k^c}$ for some small constant $c > 0$. Combining these realizations of Items 1 and 2 yields the desired $\widetilde{O}(n)$-time extractor with order-optimal seed length $O(\log(n/\varepsilon))$ and output length $(1-\eta)n$ for arbitrary constant $\eta > 0$, provided that $\varepsilon \geq 2^{-k^c}$. See Theorem 5.1 for the formal statement.

Getting around the limitation of the ideal approach. We saw above that combining the ideal approach to seeded extraction from high min-entropy sources with the new analysis of the bounded independence sampler yields a conceptually simple construction with the desired properties when the error is not too small (or alternatively, whenever the entropy guarantee is large enough). However, we would like to have $\widetilde{O}(n)$-time seeded extraction with $O(\log(n/\varepsilon))$ seed length and large output length for all ranges of parameters.

To get around this limitation of our first construction, it is natural to turn to other classical approaches for constructing nearly-optimal extractors for high min-entropy sources, such as those of Srinivasan and Zuckerman [SZ99] or Guruswami, Umans, and Vadhan [GUV09]. These approaches consist of intricate recursive procedures combining a variety of combinatorial objects, and require a careful analysis.[7] However, we could not find such an approach that works as is, even when instantiated with $\widetilde{O}(n)$-time condensers and $\widetilde{O}(n)$-time hash-based extractors. In particular:

- The GUV approach [GUV09] gives explicit seeded extractors with large output length and order-optimal seed length for *any* min-entropy requirement k and error ε. However, its overall runtime is significantly larger than $\widetilde{O}(n)$ whenever ε is not extremely small (for example, $\varepsilon = 2^{-k^\alpha}$ for some $\alpha \in (0, 1/2)$ is not small enough).

[7] In our view, these approaches are much less conceptually appealing than the "ideal" approach above. We believe that obtaining conceptually simpler constructions of fast nearly-optimal extractors that work for all errors is a worthwhile research direction, even if one does not improve on the best existing parameters.

- The SZ approach [SZ99] can be made to run in time $\widetilde{O}(n)$ and have large output length when instantiated with fast condensers, samplers, and hash-based extractors, but it is constrained to error $\varepsilon \geq 2^{-ck/\log^* n}$, where $\log^*$ is the iterated logarithm.

Fortunately, the pros and cons of the GUV and SZ approaches complement each other. Therefore, we can obtain our desired result by applying appropriately instantiated versions of the GUV and SZ approaches depending on the regime of ε we are targeting.

1.4 Future Work

We list here some directions for future work:

- Remove the preprocessing step that our constructions behind Theorem 1 require when $k < C \log^2(n/\varepsilon)$.
- On the practical side, develop concrete implementations of seeded extractors with near-optimal seed length and large output length. In particular, we think that our non-recursive construction in Section 5.1 holds promise in this direction.

1.5 Acknowledgements

Part of this research was done while the authors were visiting the Simons Institute for the Theory of Computing, supported by DOE grant # DE-SC0024124. D. Doron's research was also supported by Instituto de Telecomunicações (ref. UIDB/50008/2020) with the financial support of FCT - Fundação para a Ciência e a Tecnologia and by NSF-BSF grant #2022644. J. Ribeiro's research was also supported by Instituto de Telecomunicações (ref. UIDB/50008/2020) and NOVA LINCS (ref. UIDB/04516/2020) with the financial support of FCT - Fundação para a Ciência e a Tecnologia.

2 Preliminaries

2.1 Notation

We often use uppercase roman letters to denote sets and random variables – the distinction will be clear from context. We denote the support of a random variable X by $\mathsf{supp}(X)$, and, for a random variable X and set S, we also write $X \sim S$ to mean that X is supported on S. For a random variable X, we write $x \sim X$ to mean that x is sampled according to the distribution of X. We use U_d to denote a random variable that is uniformly distributed over $\{0,1\}^d$. For two strings x and y, we denote their concatenation by $x \circ y$. Given two random variables X and Y, we denote their product distribution by $X \times Y$ (i.e., $\Pr[X \times Y = x \circ y] = \Pr[X = x] \cdot \Pr[Y = y]$. Given a positive integer n, we write $[n] = \{1, \ldots, n\}$. For a prime power q, we denote the finite field of order q by $\mathbb{F}_q$. We denote the base-2 logarithm by log.

2.2 Model of Computation

We work in the standard, multi-tape, Turing machine model with some fixed number of work tapes. In particular, there exists a constant C such that all our claimed time bounds hold whenever we work with at most C work tapes. This also implies that our results hold in the RAM model, wherein each machine word can store integers up to some fixed length, and standard word operations take constant time. In Section 4 we will give, in addition to the standard Turing machine model bounds, an improved runtime bound that is dedicated to the logarithmic-cost RAM model.

2.3 Fast Finite Fields Operations

For a prime power $q = p^\ell$, we let $M_q(d)$ be the number of field operations required to multiply two univariate polynomials over $\mathbb{F}_q$ of degree less than d, and $M_q^{\mathsf{b}}(d)$ be the bit complexity of such a multiplication, so $M_q^{\mathsf{b}}(d) \leq M_q(d) \cdot T(q)$, where we denote by $T(q)$ an upper bound on the bit complexity of arithmetic operations in $\mathbb{F}_q$. When $\ell = 1$, Harvey and van der Hoeven [HvdH19, HvdH21] showed that

$$M_q^{\mathsf{b}}(d) = O(d \log q \cdot \log(d \log q) \cdot 4^{\max(0, \log^* d - \log^* q)}),$$

and in general, $M_q(d) = d \cdot \log d \cdot 2^{O(\log^* n)}$ [Für09].[8] When $p = 2$, we can use Schönhage's algorithm [Sch77] to get $M_q^{\mathsf{b}}(d) = O(d \log d \cdot \operatorname{loglog} d \cdot M_q(\log q))$, where we relied on the fact that addition and multiplication in $\mathbb{F}_q$ can be done in time $M_q(\ell) = O(\ell \cdot \log \ell \cdot \operatorname{loglog} \ell)$. Overall, when $d \leq q \leq 2^d$, and q is either a prime or a power of two, $M_q^{\mathsf{b}}(d) = d \log d \cdot \widetilde{O}(\log q)$. We will use fast multi-point evaluation and fast computation of derivatives (together with the preceding bounds on M_q^{b}).

Lemma 2.1 ([BM74], see also [vzGG13, Chapter 10])**.** *Let $d \in \mathbb{N}$, and let q be a prime, or a power of 2. Then, given a polynomial $f \in \mathbb{F}_q[X]$ of degree at most d, the following holds.*

1. *Given a set $\{\alpha_1, \ldots, \alpha_t\} \subseteq \mathbb{F}_q$, where $t \leq d$, one can compute $f(\alpha_1), \ldots, f(\alpha_t)$ in time $O(M_q^{\mathsf{b}}(d) \cdot \log d) = d \log^2 d \cdot \widetilde{O}(\log q)$.*
2. *For $t \leq d$ and $\alpha \in \mathbb{F}_q$, one can compute the derivatives $f(\alpha), f'(\alpha), \ldots, f^{(t)}(\alpha)$ in time $O(M_q(d) \cdot \log d) = d \log^2 d \cdot \widetilde{O}(\log q)$.*

Note that when $q \leq 2^d$, we can bound $O(M_q(d) \cdot \log d)$ by $\widetilde{O}(d) \cdot \log q$.[9]

For a comprehensive discussion of fast polynomial arithmetic, see Von Zur Gathen and Gerhard's book [vzGG13] (and the more recent important developments [HvdH21]).

2.4 Statistical Distance, Entropy

We present some relevant definitions and lemmas about the statistical distance and min-entropy.

Definition 2.2 (statistical distance)**.** *The* statistical distance *between two random variables X and Y supported on $\mathcal{S}$, denoted by $\Delta(X, Y)$, is defined as*

$$\Delta(X, Y) = \sup_{\mathcal{T} \subseteq \mathcal{S}} |\Pr[X \in \mathcal{T}] - \Pr[Y \in \mathcal{T}]| = \frac{1}{2} \sum_{x \in \mathcal{S}} |\Pr[X = x] - \Pr[Y = x]|.$$

*We say that X and Y are ε-*close*, and write $X \approx_\varepsilon Y$, if $\Delta(X, Y) \leq \varepsilon$.*

Definition 2.3 (min-entropy)**.** *The* min-entropy *of a random variable X supported on $\mathcal{X}$, denoted by $\mathbf{H}_\infty(X)$, is defined as*

$$\mathbf{H}_\infty(X) = -\log\left(\max_{x \in \mathcal{X}} \Pr[X = x]\right).$$

Above, and throughout the paper, we use base-2 logarithms.

[8] If $\mathbb{F}_q$ contains a d-th root of unity, one can get $M_q(d) = d \log d$ from the classic FFT algorithm [CT65]. For a simpler algorithm attaining the bound $M_q(d) = d \log d \operatorname{loglog} d$, see [vzGG13, Sections 8,10]. See also [HvdH22] for a widely-believed conjecture under which $M_q(d) = d \log d$ always holds.

[9] The looser bound of $\widetilde{O}(d) \cdot \log q$, when $q \leq 2^d$, is also a bound for an arbitrary $\mathbb{F}_q$, and can be achieved with simpler algorithms than the ones cited.

Definition 2.4 (average conditional min-entropy)**.** *Let X and Y be two random variables supported on $\mathcal{X}$ and $\mathcal{Y}$, respectively. The* average conditional min-entropy *of X given Y, denoted by $\widetilde{\mathbf{H}}_\infty(X|Y)$, is defined as*

$$\widetilde{\mathbf{H}}_\infty(X|Y) = -\log\Big(\mathbb{E}_{y\sim Y}[2^{-\mathbf{H}_\infty(X|Y=y)}]\Big).$$

The following standard lemma gives a chain rule for min-entropy.

Lemma 2.5 (see, e.g., [DORS08])**.** *Let X, Y, and Z be arbitrary random variables such that $|\mathsf{supp}(Y)| \le 2^\ell$. Then,*

$$\widetilde{\mathbf{H}}_\infty(X|Y,Z) \ge \widetilde{\mathbf{H}}_\infty(X|Z) - \ell.$$

We can turn the chain rule above into a high probability statement.

Lemma 2.6 (see, e.g., [MW97])**.** *Let X, Y, and Z be random variables such that $|\mathsf{supp}(Y)| \le 2^\ell$. Then,*

$$\Pr_{y\sim Y}[\widetilde{\mathbf{H}}_\infty(X|Y=y,Z) \ge \widetilde{\mathbf{H}}_\infty(X|Z) - \ell - \log(1/\delta)] \ge 1-\delta$$

for any $\delta > 0$.

Definition 2.7 (smooth min-entropy)**.** *We say that a random variable X has ε-*smooth min-entropy *at least k, denoted by $\mathbf{H}^\varepsilon_\infty(X) \ge k$, if there exists a random variable X' such that $X \approx_\varepsilon X'$ and $\mathbf{H}_\infty(X') \ge k$.*

2.5 Extractors and Condensers

Definition 2.8 ((n,k)-source)**.** *We say that a random variable X is an (n,k)-*source *if $X \sim \{0,1\}^n$ and $\mathbf{H}_\infty(X) \ge k$.*

Definition 2.9 (block source)**.** *A random variable X is an $((n_1,n_2,\ldots,n_t),(k_1,\ldots,k_t))$-*block source *if we can write $X = X_1 \circ X_2 \circ \cdots \circ X_t$, each $X_i \in \{0,1\}^{n_i}$, where $\widetilde{\mathbf{H}}_\infty(X_i|X_1,\ldots,X_{i-1}) \ge k_i$ for all $i \in [s]$. In the special case where $k_i = \alpha n_i$ for all $i \in [t]$, we say that X is an $((n_1,n_2,\ldots,n_t),\alpha)$-*block source.

We say that X is an exact *block source if $\mathbf{H}_\infty(X_i|X_1 = x_1,\ldots,X_{i-1} = x_{i-1}) \ge k_i$ for* any *prefix $x_1,\ldots,x_{i-1}$. Lemma 2.6 tells us that any $((n_1,\ldots,n_t),\alpha)$-block-source is ε-close to an exact $((n_1,\ldots,n_t),(1-\zeta)\alpha)$-block-source, where $\varepsilon = \sum_{i=1}^t 2^{-\alpha\zeta n_i}$.*

Definition 2.10 (seeded extractor)**.** *A function $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ε)* seeded extractor *if the following holds. For every (n,k)-source X,*

$$\mathsf{Ext}(X,Y) \approx_\varepsilon U_m,$$

where Y is uniformly distributed over $\{0,1\}^d$ and is independent of X and U_m is uniformly distributed over $\{0,1\}^m$. We say that Ext is strong *if $\mathsf{Ext}(X,Y) \circ Y \approx_\varepsilon U_{m+d}$.*

Furthermore, Ext is said to be an average-case (k,ε) (strong seeded) extractor *if for all correlated random variables X and W such that X is supported on $\{0,1\}^n$ and $\widetilde{\mathbf{H}}_\infty(X|W) \ge k$ we have*

$$\mathsf{Ext}(X,Y) \circ Y \circ W \approx_\varepsilon U_{m+d} \circ W,$$

where Y is uniformly distributed over $\{0,1\}^d$ and is independent of X and U_{m+d} is uniformly distributed over $\{0,1\}^{m+d}$ and independent of W.

Remark 2.11. By Lemma 2.6, every strong (k,ε)-seeded extractor $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is also an average-case strong $(k' = k + \log(1/\varepsilon), \varepsilon' = 2\varepsilon)$-seeded extractor.

Definition 2.12 (condenser)**.** *A function* $\mathsf{Cond}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a* (k, k', ε) (seeded) condenser *if the following holds. For every* (n,k)*-source* X*,* $\mathbf{H}^{\varepsilon}_{\infty}(\mathsf{Cond}(X,Y)) \geq k'$*, where* Y *is uniformly distributed over* $\{0,1\}^d$ *and is independent of* X*.*

We say that Cond *is* strong *if* $Y \circ \mathsf{Cond}(X,Y)$ *is* ε*-close to some distribution* $Y \circ D$ *with min-entropy* k' *(and note that here, necessarily,* d *bits of entropy come from the seed). Finally, we say that* Cond *is* lossless *if* $k' = k + d$*.*

We also define extractors tailored to block sources.

Definition 2.13 (block source extractor)**.** *A function* $\mathsf{BExt}\colon \{0,1\}^{n_1} \times \cdots \times \{0,1\}^{n_t} \times \{0,1\}^d \to \{0,1\}^m$ *is a* $(k_1, \ldots, k_t, \varepsilon)$ strong block-source extractor *if for any* $((n_1, n_2, \ldots, n_t), (k_1, \ldots, k_t))$*-block-source* X*,*

$$\mathsf{BExt}(X,Y) \circ Y \approx_{\varepsilon} U_{m+d},$$

where Y *is uniformly distributed over* $\{0,1\}^d$ *and is independent of* X *and* U_{m+d} *is uniformly distributed over* $\{0,1\}^{m+d}$*.*

We will also require the following extractors based on the leftover hash lemma and fast hash functions. We state a result from [TSSR11] which requires seed length $d \approx 2m$, where m is the output length. It is possible to improve the seed length to $d \approx m$, but this requires the input length n to be structured [HT16].

Lemma 2.14 (fast hash-based extractors [TSSR11, Theorem 10], adapted. See also [HT16, Table I])**.** *For any positive integers* n*,* k*, and* m *and any* $\varepsilon > 0$ *such that* $k \leq n$ *and* $m \leq k - 4\log(16/\varepsilon)$ *there exists a* (k, ε)*-strong seeded extractor* $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *with seed length* $d \leq 2(m + \log n + 2\log(1/\varepsilon) + 4)$*. Moreover,* Ext *can be computed in time* $O(n \log n)$*.*

Note that by appending the seed to the output of the extractor, we can get the following: There exists a constant c *such that for any constant* $\theta \leq \frac{1}{3}$*,* $d \geq c\log(n/\varepsilon)$ *and* $k \geq \theta d + c\log(1/\varepsilon)$*, there exists a strong* (k, ε)*-seeded extractor* $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{(1+\theta)d}$*.*

2.6 Averaging Samplers

In this section we define averaging samplers and state some useful related results and constructions.

Definition 2.15 (averaging sampler)**.** *We say that* $\mathsf{Samp}\colon \{0,1\}^r \to [n]^m$ *is a* (γ, θ)-averaging sampler *if*

$$\Pr_{(i_1,\ldots,i_m)\sim\mathsf{Samp}(U_r)}\left[\left|\frac{1}{t}\sum_{j=1}^{m} f(i_j) - \mathbb{E}[f]\right| \geq \theta\right] < \gamma$$

for every function $f\colon [n] \to [0,1]$*, where* $\mathbb{E}[f] = \frac{1}{n}\sum_{i\in[n]} f(i)$*. We say that* Samp *has* distinct samples *if* $\mathsf{Samp}(x)$ *outputs* m *distinct elements of* $[n]$ *for every input* x*. The parameter* θ *is often referred to as the* accuracy *of the sampler, and* γ *is its* confidence *parameter. Moreover, we sometimes refer to* $\mathsf{Samp}(U_r) \sim [n]^m$ *as a* (γ, θ) sampling distribution*.*

The following lemma gives guarantees on sub-sampling from a weak source using an averaging sampler.

Lemma 2.16 ([Vad04, Lemma 6.2])**.** *Let* $\delta, \gamma, \tau \in (0,1)$ *be such that* $\delta \geq 3\tau$ *and let* $\mathsf{Samp}\colon \{0,1\}^r \to [n]^m$ *be a* $(\gamma, \theta = \tau/\log(1/\tau))$*-averaging sampler with distinct samples. Then, for any* $(n, k = \delta n)$*-source* X *and* Y *uniformly distributed over* $\{0,1\}^r$ *we have that*

$$Y \circ X_{\mathsf{Samp}(Y)} \approx_{\gamma + 2^{-\Omega(\tau n)}} Y \circ W,$$

where $(W|Y = y)$ *is an* $(m, k' = (\delta - 3\tau)m)$*-source for every* y*.*

The "expander random walk" sampler. We will need the following averaging sampler based on random walks on expanders. Let G be a D-regular graph with vertex set $[n]$. We assume that the neighborhood of each vertex is ordered in some predetermined way. Then, the associated averaging sampler parses its input x as $(i_1, b_1, b_2, \ldots, b_{t-1})$, where $i_1 \in [n]$ and $b_1, \ldots, b_{t-1} \in [D]$, and outputs $\mathsf{Samp}(x) = (i_1, \ldots, i_t)$, where i_j is the b_{j-1}-th neighbor of i_{j-1} when $j > 1$. To ensure distinct samples, we skip repeated vertices.

The performance of Samp as an averaging sampler is determined by the spectral expansion of G. In fact, if G has spectral expansion $\theta/2$ then a direct application of the expander Chernoff bound [Gil98] gives that Samp is a (γ, θ)-averaging sampler with $t = O(\log(1/\gamma)/\theta^2)$ and $r = \log n + O(t \log(1/\theta))$ [Vad04, Section 8.2]. We instantiate G with the regular expander graphs from the following result of Alon [Alo21].

Lemma 2.17 ([Alo21, Theorem 1.2], adapted)**.** *Fix any prime p such that $p \equiv 1 \mod 4$. Then, there is a constant C_p such that for every integer $n \geq C_p$ there exists a $(D = p+2)$-regular graph G_n on n vertices with spectral expansion $\lambda \leq \frac{(1+\sqrt{2})\sqrt{d-1}+o(1)}{d}$, where the $o(1)$ tends to 0 as $n \to \infty$. Furthermore, the family $(G_n)_n$ is strongly explicit.*

In particular, for any $\theta > 0$ there exist constants $C_\theta > 0$ and $D_\theta = O(\theta^{-2})$ and a strongly explicit family of D_θ-regular graphs $(G_n)_{n \geq C_\theta}$ with spectral expansion $\lambda \leq \theta$ for any $n \geq C_\theta$.

Taking the t-th power of a λ-spectral expander improves its expansion to λ^t. This readily gives us the following corollary.

Corollary 2.18. *For every large enough n, and any $\lambda = \lambda(n) > 0$, there exists a D-regular graph $G = (V = [n], E)$ with spectral expansion λ, where $D = \mathrm{poly}(1/\lambda)$, and given $x \in [n]$ and $i \in [D]$, the y-th neighbor of x can be computed in time $\log(1/\lambda) \cdot \mathrm{polylog}(n)$.*

Combining the discussion above with Lemma 2.17 (or Corollary 2.18) immediately yields the following.

Lemma 2.19 ([Vad04, Lemma 8.2], appropriately instantiated)**.** *For every large enough integer n and every $\theta, \gamma \in (0,1)$, there exists a (γ, θ)-averaging sampler $\mathsf{Samp}\colon \{0,1\}^r \to [n]^t$ with distinct samples with $t = O(\log(1/\gamma)/\theta^2)$ and $r = \log n + O(t \log(1/\theta))$. Furthermore, Samp is computable in time $O(t \cdot \mathrm{polylog}\, n)$.*

We can extend Lemma 2.19 to output more distinct samples while not increasing r via the following simple lemma.

Lemma 2.20 ([Vad04, Lemma 8.3])**.** *Suppose that $\mathsf{Samp}_0\colon \{0,1\}^r \to [n]^t$ is a (γ, θ)-averaging sampler with distinct samples. Then, for every integer $m \geq 1$ there exists a (γ, θ)-averaging sampler $\mathsf{Samp}\colon \{0,1\}^r \to [m \cdot n]^{m \cdot t}$ with distinct samples. Furthermore, if Samp_0 is computable in time T, then Samp is computable in time $O(mT)$.*

Lemma 2.20 follows easily by parsing $[m \cdot t] = [m] \times [t]$ and considering the sampler $\mathsf{Samp}(x)_{i,j} = (i, \mathsf{Samp}_0(x)_j)$ for $i \in [m]$ and $j \in [t]$. If we can compute $\mathsf{Samp}_0(x)$ in time T, then we can compute $\mathsf{Samp}(x)$ in time $O(mT)$, as desired. The following is an easy consequence of Lemmas 2.19 and 2.20.

Lemma 2.21 ([Vad04, Lemma 8.4], with additional complexity claim)**.** *There exists a constant $C > 0$ such that the following holds. For every large enough n and $\theta, \gamma \in (0,1)$, there exists a (γ, θ)-averaging sampler $\mathsf{Samp}\colon \{0,1\}^r \to [n]^t$ with distinct samples for any $t \in [t_0, n]$ with $t_0 \leq C \log(1/\gamma)/\theta^2$ and $r = \log(n/t) + \log(1/\gamma) \cdot \mathrm{poly}(1/\theta)$. Furthermore, Samp is computable in time $t \cdot \mathrm{poly}(1/\theta, \log n)$.*

In particular, if θ is constant then $t_0 = O(\log(1/\gamma))$, $r = \log(n/t) + O(\log(1/\gamma))$, and Samp is computable in time $t \cdot \mathrm{polylog}\, n$.

2.7 Standard Composition Techniques for Extractors

We collect some useful classical techniques for composing seeded extractors.

Lemma 2.22 (boosting the output length [WZ99, RRV02])**.** *Suppose that for $i \in \{1,2\}$ there exist strong (k_i, ε_i)-seeded extractors $\mathsf{Ext}_i\colon \{0,1\}^n \times \{0,1\}^{d_i} \to \{0,1\}^{m_i}$ running in time T_i, with $k_2 \le k_1 - m_1 - g$. Then, $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^{d_1+d_2} \to \{0,1\}^{m_1+m_2}$ given by $\mathsf{Ext}(X,(Y_1,Y_2)) = \mathsf{Ext}_1(X,Y_1) \circ \mathsf{Ext}(X,Y_2)$ is a strong $(k_1, \frac{\varepsilon_1}{1-2^{-g}} + \varepsilon_2)$-seeded extractor running in time $O(T_1 + T_2)$.*

Lemma 2.23 (block source extraction)**.** *Let $X = X_1 \circ \cdots \circ X_t$ be an $((n_1,\dots,n_t),(k_1,\dots,k_t))$-block-source, and let $\mathsf{Ext}_i\colon \{0,1\}^{n_i} \times \{0,1\}^{d_i} \to \{0,1\}^{m_i}$ be average-case strong (k_i,ε_i)-seeded extractors running in time T_i with output length $m_i \ge d_{i-1} - d_i$ for $i \ge 2$. Then, there exists a strong $(k_1,\dots,k_t, \varepsilon = \sum_{i\in[t]} \varepsilon_i)$-block-source extractor $\mathsf{BExt}\colon \{0,1\}^{n_1} \times \cdots \times \{0,1\}^{n_t} \times \{0,1\}^{d_t} \to \{0,1\}^m$ with output length $m = m_1 + \sum_{i=2}^{t}(m_i - (d_{i-1} - d_i))$ that runs in time $O(\sum_{i\in[t]} T_i)$. If X is an exact block source, then the Ext_i's do not need to be average-case.*

We discuss how the fast hash-based extractor from Lemma 2.14 can be used to construct a fast extractor with seed length any constant factor smaller than its output length for high min-entropy sources. We need the following lemma, which is an easy consequence of the chain rule for min-entropy.

Lemma 2.24 ([GUV09, Corollary 4.16])**.** *If X is an $(n, k = n - \Delta)$-source and we write $X = X_1 \circ \cdots \circ X_t$ with $|X_i| \ge n'$ for all $i \in [t]$, then $X_1 \circ \cdots \circ X_t$ is $t\varepsilon$-close to an $(n_1 = n', \dots, n_t = n', k' = n' - \Delta - \log(1/\varepsilon))$-block-source.*

The following appears in [GUV09] without the time complexity bound. We appropriately instantiate their approach and analyze the time complexity below.

Lemma 2.25 (fast extractors with seed shorter than output [GUV09, Lemma 4.11])**.** *For every integer $t \ge 1$ there exists a constant $C > 0$ such that for any positive integer n and $\varepsilon > 2^{-\frac{n}{50t}}$ there exists a strong $(k = (1 - \frac{1}{20t})n, \varepsilon)$-seeded extractor $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with $m \ge k/2$ and $d \le k/t + C\log(n/\varepsilon)$ computable in time $O(tn\log n)$.*

Proof. Let X be an $(n, k = (1 - \frac{1}{20t})n)$-source and $\varepsilon' = \frac{\varepsilon}{2t}$. Write X as $X = X_1 \circ \cdots \circ X_t$ with $|X_i| = \lfloor n/t \rfloor = n'$ for all i. Then, Lemma 2.24 guarantees that $X_1 \circ \cdots \circ X_t$ is $(t\varepsilon')$-close to an $(n_1 = n', \dots, n_t = n', k' = n' - \frac{n}{20t} - \log(1/\varepsilon'))$-block-source X'. Note that

$$k' = n' - \frac{n}{20t} - \log(1/\varepsilon') \ge \frac{19n}{20t} - 1 - \log(1/\varepsilon') \ge 0.9n',$$

since we have assumed that $\varepsilon > 2^{-\frac{n}{50t}}$. Now, let $\mathsf{Ext}'\colon \{0,1\}^{n'} \times \{0,1\}^d \to \{0,1\}^m$ be the strong (k',ε')-seeded extractor from Lemma 2.14 with output length $m = \lceil \frac{k}{2t} \rceil \le k - 4\log(16/\varepsilon')$ and corresponding seed length $d \le k/t + 4\log(n/\varepsilon') + 9 \le k/t + C\log(n/\varepsilon)$ for a large enough constant $C > 0$ depending only on t. Then, we apply block source extraction (Lemma 2.23) to X' with $\mathsf{Ext}_1 = \cdots = \mathsf{Ext}_t = \mathsf{Ext}'$ to get the desired strong $(k, 2t\varepsilon' = \varepsilon)$-extractor Ext with output length $t \cdot m \ge k/2$ and seed length d. Since Ext' is computable in time $O(n\log n)$, then Ext is computable in time $O(tn\log n)$. □

In addition to Lemma 2.22, one can potentially boost the output length of a high min-entropy extractor by first treating the source as a block sources and then performing a simple block source extraction. The following corollary follows easily from Lemmas 2.23 and 2.24 (and can also be found in [Vad04, Section 6]).

Lemma 2.26. *Let* $\mathsf{Ext}_{\mathsf{in}}\colon \{0,1\}^{n/2}\times\{0,1\}^{\ell}\to\{0,1\}^{d}$ *and* $\mathsf{Ext}_{\mathsf{out}}\colon \{0,1\}^{n/2}\times\{0,1\}^{d}\to\{0,1\}^{m}$ *be* (k',ε)*-extractors. Then, for any* $(n,k=\delta n)$*-source* $X_1\circ X_2$*, each* $X_i\sim\{0,1\}^{n/2}$*, and an independent uniform* $Y\sim\{0,1\}^{\ell}$*, we have that*

$$\mathsf{Ext}((X_1,X_2),Y)=\mathsf{Ext}_{\mathsf{out}}(X_1,\mathsf{Ext}_{\mathsf{in}}(X_2,Y))$$

is 4ε*-close to uniform, assuming that* $k'\geq(\delta-\frac{3}{4})n$ *and* $\varepsilon\geq 2^{-n/4}$*. In other words,* Ext *is a* $(k,4\varepsilon)$*-extractor. Moreover, if* $\mathsf{Ext}_{\mathsf{in}}$ *is strong then* Ext *is also strong, and if* $\mathsf{Ext}_{\mathsf{in}}$ *and* $\mathsf{Ext}_{\mathsf{out}}$ *run in time* T_1 *and* T_2*, respectively, then* Ext *runs in time* $O(T_1+T_2)$*.*

3 Additional Building Blocks

3.1 Fast Generation of Small-Bias Sets

A set $S\subseteq\{0,1\}^n$ is *ε-biased* if the uniform distribution over its elements is indistinguishable from uniform by every linear test. Namely, if for every nonempty $T\subseteq[n]$ it holds that $\Pr_{s\sim S}[\bigoplus_{i\in T}s_i]\in[\frac{1-\varepsilon}{2},\frac{1+\varepsilon}{2}n]$. We say that a linear code $\mathcal{C}\subseteq\{0,1\}^n$ is ε-balanced if the Hamming weight of each nonzero codeword lies in $[\frac{1-\varepsilon}{2}n,\frac{1+\varepsilon}{2}]$. It is known that these two objects are essentially the same: S is ε-biased if and only if the $|S|\times n$ matrix whose rows are the elements of S is a generator matrix of an ε-balanced code.

One prominent way of constructing ε-balanced codes is via *distance amplification*, namely, starting with a code of some bias $\varepsilon_0\gg\varepsilon$ and, using a parity sampler, amplify its bias. We will use a specific, simple, instantiation of a parity sampler – the random walk sampler.

Lemma 3.1 (RWs amplify bias [Ta-17][10])**.** *Let* $\mathcal{C}_0\subseteq\{0,1\}^n$ *be an* ε_0*-balanced code, and let* $G=(V=[n],E)$ *be a* D*-regular* λ*-spectral expander, and for an even* $t\in\mathbb{N}$*, let* $\mathcal{W}_t=\{w_1,\ldots,w_{\bar{n}}\}\subseteq[n]^t$ *be the set of walks of length* t *on* G*, noting that* $\bar{n}=n\cdot D^t$*. Define* $\mathcal{C}\subseteq\{0,1\}^{\bar{n}}$ *such that*

$$\mathcal{C}=\{\mathrm{dsum}_{\mathcal{W}_t}(c_0):c_0\in\mathcal{C}_0\},$$

where $y=\mathrm{dsum}_{\mathcal{W}_t}(x)$ *at location* $i\in[\bar{n}]$ *is given by* $\bigoplus_{j\in w_i}x_j$*.*

Then, $\mathcal{C}$ *is* ε*-balanced, for*

$$\varepsilon=(\varepsilon_0+2\lambda)^{t/2}.$$

For $\mathcal{C}_0$, we will use the Justesen code.

Lemma 3.2 ([Jus72])**.** *There exist constants* $R\in(0,1)$ *and* $\varepsilon_0\in(0,1)$ *such that there exists an explicit family of codes* $\{\mathrm{Jus}_n\}$*, each of which has block length* n*, rate* R*, and is* ε_0*-balanced. Moreover, given* $x\in\{0,1\}^{k=Rn}$*,* $\mathrm{Jus}_n(x)$ *can be computed in* $\widetilde{O}(n)$*.*

Proof. The parameters of the codes follow from the original construction (and specifically, the lemma holds, say, with $R=\frac{1}{8}$ and $\varepsilon_0=\frac{37}{40}$), so we will just show that the code is efficiently computable. Given a message x, we first encode it with a full Reed–Solomon code of constant relative distance over a field $\mathbb{F}_q$ of characteristic 2, where $q\log q=O(n)$. By Lemma 2.1, this can be done in time $\widetilde{O}(q)=\widetilde{O}(n)$. Then, we encode with polynomial evaluation $p_x(\alpha)$, for $\alpha\in\mathbb{F}_q$, with the binary representation of $(p(\alpha),\alpha\cdot p(\alpha))$. This takes $\widetilde{O}(q)$ time as well. □

[10] The argument for $t=2$ was suggested already by Rozenman and Wigderson (see [Bog12]).

Corollary 3.3. *There exist a constant $c > 1$, and an explicit family of balanced codes, such that for every $\bar{n} \in \mathbb{N}$ and any $\varepsilon > 0$, $\mathcal{C} \subseteq \mathbb{F}_2^{\bar{n}}$ is ε-balanced of rate $R = \varepsilon^c$, and given $x \in \mathbb{F}_2^{k=R\bar{n}}$, any m bits of $\mathcal{C}(x)$ can be computed in time $\widetilde{O}(n) + O(m \log(1/\varepsilon) \log n \operatorname{loglog} n)$.*

Moreover, for every $k \in \mathbb{N}$ and any $\varepsilon > 0$ there exists an explicit ε-biased set over $\mathbb{F}_2^k$ generated by a function $\mathsf{SmallBias}\colon [\bar{n}] \to \{0,1\}^k$ computable in time $(k + \log(1/\varepsilon)) \cdot \widetilde{O}(\log k)$.

Proof. Let $\mathcal{C}_0\colon \mathbb{F}_2^{k=\Theta(n)} \to \mathbb{F}_2^n$ be the ε_0-balanced code guaranteed to us by Lemma 3.2, and let $G = (V = [n], E)$ be the D-regular λ-spectral expander of Corollary 2.18, instantiated with $\lambda = \frac{1-\varepsilon_0}{4}$ (so $D = D(\varepsilon_0)$). Letting $\mathcal{C}\colon \mathbb{F}_2^k \to \mathbb{F}_2^{\bar{n}}$ be the amplified code of Lemma 3.1 set with

$$t = \frac{2\log(1/\varepsilon)}{\log\left(\frac{1+\varepsilon_0}{2}\right)} = O(\log(1/\varepsilon)),$$

the lemma tells us that it is $(\varepsilon_0 + 2\lambda)^{t/2} \leq \varepsilon$ balanced. Given $x \in \mathbb{F}_2^n$ and $i \in [\bar{n}]$, computing $\mathcal{C}(x)_i$ amounts to XORing t coordinates of $\mathcal{C}_0(x)$ determined by $i = (v, i_1, \ldots, i_t)$, which indexes a random walk over G. Computing $\mathcal{C}_0(x)$ takes $\widetilde{O}(n)$ time, and computing a length-t walk over G takes $t \cdot O(\log(1/\lambda) \cdot \log n \cdot \operatorname{loglog} n)$ time. The corollary then follows, observing that $\bar{n} = n \cdot D^t = n \cdot \operatorname{poly}(1/\varepsilon)$, and that

$$\widetilde{O}(n) + m \cdot t \cdot O(\log n \cdot \operatorname{loglog} n) = \widetilde{O}(n) + m \cdot \log(1/\varepsilon).$$

For the "Moreover" part, recall that we can take the rows of the generator matrix of $\mathcal{C}$ as our ε-biased set S. Thus, for any $i \in [\bar{n}]$, we can compute $\mathsf{SmallBias}(i)$ as follows: Compute the corresponding random walk on G, and then, for any $j \in [k]$, $\mathsf{SmallBias}(i)_j$ is obtained by XORing the bits of $\mathcal{C}_0(e_j)$ indexed by the i-th random walk. Observing that each bit of $\mathcal{C}_0(e_j)$ can be computed in time $\widetilde{O}(\log n)$,[11] the runtime of $\mathsf{SmallBias}$ is

$$t \cdot O(\log(1/\lambda) \cdot \log n \cdot \operatorname{loglog} n) + k \cdot \widetilde{O}(\log n) = (k + \log(1/\varepsilon)) \cdot \widetilde{O}(\log k). \qquad \square$$

Remark 3.4. Instead of using Justesen's code from Lemma 3.2 as our inner code $\mathcal{C}_0$, we can instead use the linear-time encodable code of Spielman [Spi96]. While not stated as *balanced* codes, but rather as constant relative distance codes, one can verify that the distance can also be bounded by above. The construction is more involved than Justesen's one. However, in the logarithmic-cost RAM model, in which basic register operations over $O(\log n)$ bit registers count as a single time step, Spielman's code can be implemented in $O(n)$ time.

3.2 A Sampler from Bounded Independence

Recall that $X_1, \ldots, X_n \sim \Sigma^n$ is a (b, ε)-wise independent distribution, if for every distinct $i_1, \ldots, i_b \in [n]$ it holds that $(X_{i_1}, \ldots, X_{i_b})$ is ε-close to the uniform distribution over Σ^b. Given our efficiently generated small biased spaces, we can efficiently generate almost b-wise independent sample spaces as well.

Lemma 3.5. *For any positive integers n, $m \leq n$, and $b \leq m$, and any $\varepsilon > 0$, there exists an explicit (b, ε)-wise independent generator $\mathsf{BI}_{b,\varepsilon}\colon \{0,1\}^d \to [n]^m$ with $d = O(b \log n + \log(1/\varepsilon))$. That is, the distribution formed by picking $z \sim U_d$ and outputting $\mathsf{BI}_{b,\varepsilon}(z)$ is (b, ε)-wise independent over $[n]^m$. Moreover,*

1. *Given $z \in \{0,1\}^d$, $\mathsf{BI}_{b,\varepsilon}(z)$ is computable in time $\widetilde{O}(n)$.*

[11] Indeed, each coordinate of $\mathcal{C}_0(e_j)$ is a bit in the encoding of (α^j, α^{j+1}) for some $\alpha \in \mathbb{F}_q$, where $q \log q = O(n)$.

2. *Assume that $\theta \in (0, 1/4)$ is such that $\varepsilon \leq \theta \cdot n^{-b/2}$. Then, with probability at least $1 - 2^{-\Omega(\theta b)}$ over $z \sim U_d$, $\mathsf{Bl}_{b,\varepsilon}(z)$ has at least $m - (1+4\theta)\frac{m^2}{n}$ distinct elements.*

Proof. Let $q = 2^{\lceil \log n \rceil}$, and set $\gamma = \varepsilon \cdot 2^{-\frac{b \log q}{2}}$ and $n_{\mathsf{b}} = n \log q$. Let $S_{\mathsf{b}} \subseteq \{0,1\}^{n_{\mathsf{b}}}$ denote the γ-biased set that is guaranteed to us by Corollary 3.3 and let $S \subseteq [q]^n$ be the set that corresponds to treating each consecutive $\log q$ bits as an element of $[q]$.[12] By the XOR lemma, S is (b,ε)-biased over $[q]^n$. Moreover, $\log |S| = O(\log(1/\varepsilon) + k \log q + \log n) = O(\log(1/\varepsilon) + b \log n)$, and each element of S can be generated in time

$$(n_{\mathsf{b}} + \log(1/\gamma)) \cdot \widetilde{O}(\log n_{\mathsf{b}}) = \widetilde{O}(n) + (b + \log(1/\varepsilon)) \cdot \widetilde{O}(\log n) = \widetilde{O}(n),$$

where the last equality follows since we can always assume that $\varepsilon \geq m^{-n}$. Notice that ignoring the last $n - m$ symbols of each element of S still preserves the above properties, which indeed gives rise to an efficiently samplable (b,ε)-wise independent sample space over $[q]^m$.

Next, we argue that most samples contain mostly distinct elements. Towards this end, let $X_1, \ldots, X_m$ be our (b,ε)-wise independent distribution $\mathsf{Bl}_{b,\varepsilon}(U_d)$, and let Z_i denote the indicator random variable that is 1 if and only if X_i is a duplicate element (namely, there exists $j < i$ such that $X_i = X_j$). We are looking to bound $\sum_{i \in [m]} Z_i$ with high probability.

Claim 3.6. *Assume that $t \leq b/2$ and $\varepsilon \leq \theta q^{-t}$ for some $\theta > 0$. Then, for any distinct $i_1, \ldots, i_t \in [m]$, it holds that $\Pr[Z_{i_1} = \ldots = Z_{i_t} = 1] \leq (1+\theta)(m/q)^t$.*

Proof. Fix indices $j_1, \ldots, j_t$, where each $j_\ell < i_\ell$. The probability that $X_{i_\ell} = X_{j_\ell}$ for all $\ell \in [t]$ is at most $q^{-t} + \varepsilon \leq (1+\theta) q^{-t}$, since this event depends on at most $2t \leq b$ random variables. Union-bounding over all choices of j-s incurs a multiplicative factor of $\prod_{\ell \in [t]} (i_\ell - 1) \leq m^t$, so overall, $\Pr[Z_{i_1} = \ldots = Z_{i_t} = 1] \leq (1+\theta)(m/q)^t$. □

Now, Claim 3.6 is sufficient to give us good tail bounds (see, e.g., [HH15, Section 3]). In particular, denoting $\mu = (1+\theta)\frac{m}{q}$ there exists a universal constant $c > 0$ such that

$$\Pr\left[\sum_{i \in [m]} Z_i \geq (1+\theta)\mu m\right] \leq 2^{-c\theta b},$$

which implies Item 2 when $n = q$. Finally, we need to argue that we can also handle the case where n is not a power of 2 (and so $q > n$). In this case, we can take our γ-biased set to be over $n_{\mathsf{b}} = \lceil \varepsilon^{-1} \log n \rceil n$ bits, and each consecutive $\lceil \varepsilon^{-1} \log n \rceil$ bits are mapped to $[n]$ by simply taking the corresponding integer modulo n. The correctness can be found, e.g., in [Rao07]. □

Towards introducing our sampler, we will need the following tail bound for (b,ε)-wise independent random variables.

Lemma 3.7 ([XZ24])**.** *Let $X \sim \Sigma^m$ be a (b,γ)-wise independent distribution, and fix some $\varepsilon > 0$. Then, X is also a (δ,ε) sampling distribution, where*

$$\delta = \left(\frac{5\sqrt{b}}{\varepsilon\sqrt{m}}\right)^b + \frac{\gamma}{\varepsilon^b}.$$

[12] Since we won't care about optimizing the dependence on n, we do not pre-encode using a bounded-independence generator (as in, say, [NN90, AGHP92]).

While the error in Item 2 above is small, it is not small enough for us to simply combine Lemmas 3.5 and 3.7, and we will need to do a mild error reduction. We do this via random walks on expanders and discarding repeating symbols, as was also done in [Vad04]. This gives us the following bounded-independence based sampler.

Lemma 3.8. *For any positive integers $m \le n$, any $\delta_\Gamma \in (0,1)$, and any constant $\eta \in (0,1)$ such that $m \le \frac{\eta}{8} n$, there exists an explicit $(\delta_\Gamma, \varepsilon_\Gamma = 2\eta)$ sampler $\Gamma\colon \{0,1\}^d \to [n]^m$ with $d = O\left(\frac{\log n}{\log m} \cdot \log \frac{1}{\delta_\Gamma}\right)$, that satisfies the following additional properties.*

1. *Every output of Γ contains distinct symbols of $[n]$, and,*
2. *Given $y \in \{0,1\}^d$, $\Gamma(y)$ is computable in time $\widetilde{O}(n + \log^2 \frac{1}{\delta_\Gamma} \cdot \frac{\log n}{\log m})$.*

Proof. Set b to be the smallest integer such that $b \log \frac{\eta\sqrt{m}}{5\sqrt{b}} \ge \log \frac{8}{\delta_\Gamma}$, set $m' = (1+\eta)m$, $\theta = \eta/4$, and $\gamma = \min\left\{\frac{1}{8}\eta^b \cdot \delta_\Gamma, \theta \cdot n^{-b/2}\right\}$. Notice that $b = O\left(\frac{\log(1/\delta_\Gamma)}{\log m}\right)$ and $\log \frac{1}{\gamma} = O\left(\frac{\log n}{\log m} \cdot \log \frac{1}{\delta_\Gamma}\right)$. Let

$$\mathsf{BI}_{b,\gamma}\colon \{0,1\}^{d'} \to [n]^{m'}$$

be the (b,γ)-wise independent generator guaranteed to us by Lemma 3.5, with $d' = O(b \log n + \log(1/\gamma)) = O(\log(1/\gamma))$. By Lemma 3.7, $X = \mathsf{BI}_{b,\varepsilon}(U_{d'})$ is a $(\delta_{\mathsf{b}}, \eta)$ sampling distribution, where

$$\delta_{\mathsf{b}} = \left(\frac{5\sqrt{b}}{\eta\sqrt{m'}}\right)^b + \frac{\gamma}{\eta^b} \le \frac{\delta_\Gamma}{8} + \frac{\delta_\Gamma}{8} \le \frac{\delta_\Gamma}{4}.$$

Also, we know from Lemma 3.5 that with probability at least $1 - 2^{-\Omega(\theta b)} \triangleq 1 - p$, each sample from X has at least $m' - (1+4\theta)\frac{m'^2}{n} \ge m$ distinct symbols, using the fact that $\frac{n}{m} \ge \frac{(1+\eta)^3}{\eta}$. Conditioned on seeing at least m distinct symbols, X as a sampling distribution, when we remove the non-distinct elements, has confidence $\frac{\delta_\Gamma/4}{1-p} \le \frac{\delta_\Gamma}{2}$ and accuracy 2η (where the second η comes from the fact that ηm symbols were removed).

Next, in order to improve the probability of sampling a good sequence, let $G = (V = \{0,1\}^{d'}, E)$ be the D-regular λ-spectral expander of Corollary 2.18, instantiated with $\lambda = p$, so $D \le p^{-c}$ for some universal constant c. Write $d = d' + \ell'$ for $\ell' = \ell \cdot \log D$, where $\ell = c_\ell \cdot \frac{\log(1/\delta_\Gamma)}{b}$ for some constant c_ℓ soon to be determined. Given $y = (z, w) \in \{0,1\}^{d'} \times [D]^\ell$, let $z = v_0, v_2, \ldots, v_\ell$ denote the corresponding random walk over G. Our sampler Γ, on input y, computes $\mathsf{BI}_{b,\gamma}(v_i)$ and outputs the first sequence with at least m distinct symbols. If no such sequence was found, Γ simply outputs $(1, \ldots, m)$ (in which case we say it *failed*). By the expander hitting property (see, e.g., [Vad12, Section 4]), Γ fails with probability at most

$$(p+\lambda)^\ell = (2p)^\ell \le \frac{\delta_\Gamma}{2}$$

over $y \sim U_d$, upon choosing the appropriate constant $c_\ell = c_\ell(\eta)$. We then have that $\Gamma(U_d)$ is indeed a $(\delta_\Gamma, 2\eta)$ sampling distribution, that can be generated using a seed of length $d' + \ell' = O(\log(1/\gamma))$. In terms of runtime, computing $v_1, \ldots, v_\ell$ can be done in time

$$\ell \cdot \log \frac{1}{p} \cdot \widetilde{O}(d') = \widetilde{O}\left(\log^2 \frac{1}{\delta_\Gamma} \cdot \frac{\log n}{\log m}\right),$$

and computing the sequences themselves takes $\ell \cdot \widetilde{O}(n)$ time. Observing that $\ell = O(\log m)$, the proof is concluded. □

We will need to somewhat extend Lemma 3.8 and use the simple, yet crucial, property of our bounded independence sampling: A subset of the coordinates of a (b,ε)-wise independent distribution with distinct samples is itself a (b,ε)-wise independent distribution with distinct samples.[13] Thus, if we wish to sample multiple times, say using $m_1 \leq \ldots \leq m_t < n$ samples, we can use *one sample* from a sampler that outputs m_t coordinates, and truncate accordingly to create the other samples. We only need to note that: (1) the sampling parameters are determined by the different m_i-s (and in particular, m_1 should be large enough), and (2) m_t needs to be small enough compared to n, so that we can get enough distinct symbols. We summarize this observation in the next lemma.

Lemma 3.9. *For any positive integers n and $m_1 < \ldots < m_t \leq n$, any $\delta \in (0,1)$ and any constant ε such that $m_t \leq \frac{\varepsilon}{16} n$, there exists an explicit function $\Gamma\colon \{0,1\}^d \to [n]^{m_t}$ with $d = O\left(\frac{\log n}{\log m_1} \cdot \log \frac{1}{\delta_\Gamma}\right)$ that satisfies the following.*

1. *For any $i \in [t]$, the function Γ_i, that on input $y \in \{0,1\}^d$ outputs $\Gamma(y)|_{[1,m_i]}$, is a (δ,ε) sampler, and each sample contains distinct symbols.*
2. *On input $y \in \{0,1\}^d$, $\Gamma(y)$ can be computed in time $\widetilde{O}(n + \log^2(1/\delta))$.*

3.3 Nearly-Linear Time Condensers

We first give the condenser based on multiplicity codes, due to Kalev and Ta-Shma [KT22].

Theorem 3.10 (the lossless KT condenser, [KT22])**.** *For any constant $\alpha \in (0,1)$ the following holds, for every $n \in \mathbb{N}$, and any $0 < \varepsilon \leq \frac{1}{n}$ and $k \geq \frac{256}{\alpha^2} \log^2 \frac{n}{\varepsilon}$. There exists an explicit strong $(k, k' = k + \ell, \varepsilon)$-condenser*

$$\mathsf{KTCond}\colon \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^m$$

where $\ell = \left(1 + \frac{1}{\alpha}\right) \log \frac{nk}{\varepsilon} = O_\alpha(\log \frac{1}{\varepsilon})$ and $m = (1+\alpha)k$. Note that the output entropy rate satisfies $\frac{k'}{m} \geq 1 - \alpha$. Moreover, given $x \in \{0,1\}^n$ and $y \in \{0,1\}^\ell$, the output $\mathsf{KTCond}(x,y)$ can be computed in $\widetilde{O}(n)$ time.

In particular, if $\varepsilon' = \sqrt{\varepsilon}$, then for all (n,k)-sources X and a $(1-\varepsilon')$-fraction of seeds y it holds that $\mathsf{KTCond}(X,y) \approx_{\varepsilon'} Z_y$, where Z_y is an $(m = (1+\alpha)k, k' - \ell = k \geq (1-\alpha)m)$-source. Note that the seed length is $\ell = O_\alpha(\log \frac{1}{\varepsilon'})$.

Proof. For the first part of the theorem statement we only need to establish the construction's runtime. Given $x \in \{0,1\}^n$ and $y \in \{0,1\}^\ell$, set a prime $p = \mathrm{poly}_\alpha(\frac{n}{\varepsilon})$,[14] and interpret x as a polynomial $f_x \in \mathbb{F}_q[X]$ of degree at most $d-1$, and y as an element of $\mathbb{F}_p$. Thus, $n = d \log q$, and we can safely ignore rounding issues, which can easily be addressed. The output $\mathsf{KTCond}(x,y)$ is the sequence of derivatives

$$\left(f(y), f'(y), \ldots, f^{(m')}(y)\right),$$

where $m' = \frac{m}{\log q}$. By Lemma 2.1, computing the derivatives takes $\widetilde{O}(d) \cdot \log p = \widetilde{O}(n)$ time. The rest of the auxiliary operations are negligible compared to computing the derivatives.

To see the "In particular" part of the theorem statement, fix an (n,k)-source X and note that $Y \circ \mathsf{KTCond}(X,Y) \approx_\varepsilon Y \circ Z$ for some Z such that $\mathbf{H}_\infty(Y \circ Z) \geq k'$. Let $Z_y = (Z|Y=y)$. Then, an averaging argument gives that for a $(1-\sqrt{\varepsilon})$-fraction of seeds y we have $\mathsf{RSCond}(X,y) \approx_{\sqrt{\varepsilon}} Z_y$. Since Y is uniformly random over $\{0,1\}^\ell$, we get that $\mathbf{H}_\infty(Z_y) \geq k' - \ell$, as desired. □

[13] We note that the distinct-samples sampler given in [Vad04] does not seem to enjoy a similar property.

[14] More precisely, they set $h = (2nk/\varepsilon)^{1/\alpha}$, and take p to be a prime between $\frac{1}{2}h^{1+\alpha}$ and $h^{1+\alpha}$.

The downside of Theorem 3.10 is that it requires the entropy in the source to be $\Omega(\log^2(1/\varepsilon))$, instead of the optimal $\Omega(\log(1/\varepsilon))$. Instead, we can use a lossy condenser[15] based on Reed–Solomon codes. Unfortunately, this comes at the expense of computing a generator of a field of size $\mathrm{poly}(1/\varepsilon)$, which we do not know how to do in nearly-linear time for arbitrary ε-s. We consider it a one-time *preprocessing* step, as it does not depend on the inputs to the condenser.

Theorem 3.11 (the lossy RS condenser, [GUV09])**.** *For any constant $\alpha \in (0,1)$ the following holds, for every $n \in \mathbb{N}$, and any $0 < \varepsilon \leq \frac{1}{n}$ and $k \geq (2+\alpha)\log(1/\varepsilon)$. There exists an explicit strong (k, k', ε)-condenser*

$$\mathsf{RSCond}\colon \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^m$$

where $\ell = \left(1+\frac{1}{\alpha}\right)\log\frac{nk}{\varepsilon} = O_\alpha(\log\frac{1}{\varepsilon})$, $m = k$, and $k' = \frac{k-\log(1/\varepsilon)}{1+\alpha} + \ell \geq (1-\alpha)k$. Note that the output entropy rate satisfies $\frac{k'}{m} \geq 1-2\alpha$. Moreover, given $x \in \{0,1\}^n$, $y \in \{0,1\}^\ell$, and a primitive element for $\mathbb{F}_{2^\ell}$, the output $\mathsf{RSCond}(x,y)$ can be computed in time $\widetilde{O}(n)$.

In particular, if $\varepsilon' = \sqrt{\varepsilon}$ and $k \geq \frac{\log(1/\varepsilon)}{\alpha(1+2\alpha)}$, then for all (n,k)-sources X and a $(1-\varepsilon')$-fraction of seeds y it holds that $\mathsf{RSCond}(X,y) \approx_{\varepsilon'} Z_y$, where Z_y is an $(m = k, k' - \ell \geq (1-2\alpha)m)$-source. Note that the seed length is $\ell = O_\alpha(\log\frac{1}{\varepsilon'})$.

Proof. We set $q = 2^\ell$, and let $\zeta \in \mathbb{F}_q$ be the generator of the multiplicative group $\mathbb{F}_q^\star$ given to us as input.[16] Given $x \in \{0,1\}^n$ and $y \in \{0,1\}^\ell$, similarly to Theorem 3.10, interpret x as a univariate polynomial f_x of degree at most $d-1$, and y as an element of $\mathbb{F}_q$. The output $\mathsf{Cond}(x,y)$ is the sequence of evaluations

$$\Big(f(y), f(\zeta y), \ldots, f(\zeta^{m'} y)\Big),$$

where $m' = \frac{m}{\log q}$.

The correctness proof, as well as the exact choice of parameters, are given in [GUV09, Section 6], so we proceed to bounding the runtime. Towards that end, since we rely on a specific primitive element ζ, we assume that the irreducible polynomial used to construct $\mathbb{F}_q$ is known, either (and there are several). Computing the evaluation points $y, \zeta y, \ldots, \zeta^{m'} y$ can then be done naively in time $m' \cdot M_q^{\mathsf{b}}(\ell) = \widetilde{O}(n)$. Then, using Lemma 2.1, the evaluation can be done in time $\widetilde{O}(d) \cdot \log q = \widetilde{O}(n)$ as well.

The "In particular" part of the theorem statement follows analogously to that of Theorem 3.10, using also the fact that if $k \geq \frac{\log(1/\varepsilon)}{\alpha(1+2\alpha)}$, then $k' - \ell = \frac{k-\log(1/\varepsilon)}{1+\alpha} \geq (1-2\alpha)k = (1-2\alpha)m$. □

4 A Faster Instantiation of Trevisan's Extractor

We first recall Trevisan's extractor [Tre01, RRV02], $\mathsf{Tre}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, set to some designated error $\varepsilon > 0$. We will need the notion of weak designs, due to Raz, Reingold, and Vadhan [RRV02].

Definition 4.1 (weak design)**.** *A collection of sets $S_1, \ldots, S_m \subseteq [d]$ is an (ℓ, ρ)-*weak design *if for all $i \in [m]$ we have $|S_i| = \ell$ and*

$$\sum_{j<i} 2^{|S_i \cap S_j|} \leq \rho(m-1).$$

[15] Our extractor will lose a small constant fraction of the entropy, so losing a small constant fraction of the entropy in the condensing step will not make much difference.

[16] Working with fields of characteristic 2 is not necessary, but may help in efficiently computing ζ. For example, Shoup [Sho90] showed that given an irreducible polynomial $f \in \mathbb{F}_2[X]$ of degree at most $d-1$, there exists a primitive element $h \in \mathbb{F}_2[X]/\langle f\rangle$ of $\mathbb{F}_2[X]/\langle f\rangle \equiv \mathbb{F}_{2^d}$ such that h is a monic polynomial of degree $O(\log d)$.

We will also need a δ-balanced code $\mathcal{C}\colon \{0,1\}^n \to \{0,1\}^{\bar{n}}$. The parameters of the weak design affect the extractor's parameters and can be set in a couple of different ways. The parameter ℓ is set to be $\log \bar{n}$, typically ρ is chosen according to m, ε, and the desired entropy k, and then d is chosen as a function of ℓ, m, and ρ according to the weak design (see [RRV02]). Given $x \in \{0,1\}^n$ and $y \in \{0,1\}^d$, Trevisan's extractor outputs

$$\mathsf{Tre}(x,y) = \bar{x}|_{y_{S_1}} \circ \ldots \circ \bar{x}|_{y_{S_m}}, \tag{1}$$

where we denote $\bar{x} = \mathcal{C}(x)$ and interpret each length-$\log \bar{n}$ bit-string y_{S_i} as a location in $[\bar{n}]$. For the runtime analysis, it will be important to recall that δ is set to be $\frac{\varepsilon}{cm}$ for some universal constant c.

Theorem 4.2. *Trevisan's extractor of Equation (1), set to extract m bits with any error $\varepsilon > 0$, is computable in time $\widetilde{O}(n + m\log(1/\varepsilon))$.*

On a RAM in the logarithmic cost model, Trevisan's extractor is computable in time $O(n) + m\log(1/\varepsilon)\cdot \mathrm{polylog}(n)$ with a preprocessing time of $\widetilde{O}(m\log(n/\varepsilon))$. In particular, there exists a universal constant c, such that whenever $m \le \frac{n}{\log^c(n/\varepsilon)}$, it runs in time $O(n)$, without the need for a separate preprocessing step.

Proof. Looking at Equation (1), note that we only need to compute m coordinates of $\mathcal{C}(x)$. To compute those m coordinates, $y_{S_1}, \ldots, y_{S_m}$, we first need to compute the weak design itself. Note that this can be seen as a preprocessing step, since it only depends on the parameters of the extractor, and not on x or y. We will use the following result.

Claim 4.3 ([FYEC24], Section A.5)**.** *For every $\ell, m \in \mathbb{N}$ and $\rho > 1$, there exists an (ℓ,ρ)-weak design $S_1, \ldots, S_m \subseteq [d]$ with $d = O(\frac{\ell^2}{\log \rho})$, computable in time $\widetilde{O}(m\ell)$.*

Once we have our preprocessing step, we are left with computing the code. By Corollary 3.3, we can choose $\bar{n}$ so that $n/\bar{n} = \delta^c$ for some universal constant c, and so $\bar{n} = n \cdot \mathrm{poly}(m, 1/\varepsilon)$ and $\ell = \log \bar{n} = O(\log(n/\varepsilon))$. Generating the design can then be done in time $\widetilde{O}(m\log(n/\varepsilon))$. Now, Corollary 3.3 tells us that any m bits of $\mathcal{C}(x)$ can be computed in time

$$\widetilde{O}(n) + O(m\log(1/\delta)\log n \log\log n) = \widetilde{O}(n + m\log(1/\varepsilon)).$$

On a RAM in the logarithmic cost model, we can use the variant of $\mathcal{C}$ that uses Spielman's code as a base code (see Remark 3.4) and get a runtime of $O(n) + m\log(1/\varepsilon)\cdot\mathrm{polylog}(n)$. This gives a truly linear time construction whenever m is at most $\frac{n}{\log(1/\varepsilon)\,\mathrm{polylog}(n)}$. □

We conclude by noting that there is a natural setting of parameters under which Trevisan's extractor gives logarithmic seed and linear (or near-linear) time. When $m = k^{\Omega(1)}$, the parameters can be set so that $d = O\left(\frac{\log^2(n/\varepsilon)}{\log k}\right)$. We thus have the following corollary.

Corollary 4.4. *For every $n \in \mathbb{N}$, any constant $c > 1$, and any constants $\alpha, \beta \in (0,1)$, Trevisan's extractor $\mathsf{Tre}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ can be instantiated as a $(k = n^\alpha, \varepsilon = n^{-c})$ extractor with $d = O(\log n)$, $m = k^\beta$, and given $x \in \{0,1\}^n$ and $y \in \{0,1\}^d$, $\mathsf{Tre}(x,y)$ is computable in time $\widetilde{O}(n)$ (or $O(n)$ time, depending on the model).*

5 Nearly-Linear Time Extractors with Order-Optimal Seed Length

5.1 A Non-Recursive Construction

In this section, we use the various previously introduced building blocks to construct a seeded extractor with order-optimal seed length $O(\log(n/\varepsilon))$ computable in time $\widetilde{O}(n)$. In a nutshell, our extractor proceeds as follows on input an (n,k)-source X:

1. Using a fresh seed, apply the lossless KT condenser from Theorem 3.10 to X. This yields an (n', k)-source X' of length $n' \approx k$ and constant entropy rate δ which can be arbitrarily close to 1.

2. Using the fact that X' has high min-entropy rate, use the bounded-independence sampler from Lemma 3.9 to sample subsources from X' using a fresh seed. Specific properties of the bounded-independence sampler allow us to obtain a block source $Z = Z_1 \circ Z_2 \circ \cdots \circ Z_t$ with a seed of length only $O(\log(1/\varepsilon))$. The number of blocks is $t = O(\log n)$ and the blocks Z_i have geometrically *increasing* lengths, up to an n^α length threshold.

3. Now, to prepare for the hash-based iterative extraction, we need to make our blocks *decreasing*. Again, using a short seed, of length $O(\log(n/\varepsilon))$, we transform Z into $S = S_1 \circ \cdots S_t$, where the blocks are now geometrically decreasing. The blocks lengths will vary from n^{β_1} to some n^{β_2}, for some constants $\beta_1 > \beta_2$.

4. Using a fresh seed, apply the fast hash-based extractor from Lemma 2.14 to perform block source extraction from S. Noting that the first block has length $n^{\Omega(1)}$, the block source extraction only outputs $n^{\Omega(1)}$ bits. We are able to use only $O(\log(n/\varepsilon))$ random bits here, since we do not output $n^{\Omega(1)}$ bits already at the beginning of the iterative extraction process, but instead output logarithmically many bits, and gradually increasing the output length.

These steps will culminate in the following theorem.

Theorem 5.1 (non-recursive construction)**.** *There exists a constant $c \in (0,1)$ such that for every positive integers n and $k \le n$, any $\varepsilon \ge 2^{-k^c}$, and any constant $\eta \in (0,1)$, there exists a strong (k, ε) extractor*

$$\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m,$$

where $d = O(\log(n/\varepsilon))$, and $m = (1-\eta)k$. Moreover, given inputs $x \in \{0,1\}^n$ and $y \in \{0,1\}^d$, we can compute $\mathsf{Ext}(x,y)$ in time $\widetilde{O}(n)$.

5.1.1 Item 2: Generating the block source

Because of the initial condensing step, we will assume from here onwards that our input source X is an $(n, k = \delta n)$-source with constant δ. In order to generate the desired block source, we first use a fresh seed Y as input to an appropriate instantiation of the bounded-independence sampler Γ from Lemma 3.9. This yields a tuple of coordinates $\Gamma(Y) = j_1, \ldots, j_{m_t}$ from $[n]$, such that $\Gamma(Y)|_{[1,m_i]}$ is an appropriate averaging sampler for every i. Then, we use these coordinates to sample subsources from $X \sim \{0,1\}^n$, and get a block source with *increasing* blocks.

Lemma 5.2 (sampling a block source)**.** *There exists a deterministic procedure that given an (n,k)-source X with $k \ge \delta n$, δ being constant, and:*

- *A constant loss parameter $\zeta \in (0,1)$,*
- *A closeness parameter $\varepsilon \in (0,1)$ that satisfies $\varepsilon \ge 2^{-c_\varepsilon n}$ where $c_\varepsilon = c_\varepsilon(\zeta, \delta)$ is constant,*
- *Number of desired blocks $t \in \mathbb{N}$,*
- *A final, maximal, block length $\Delta_t \le c_{\mathsf{t}} \cdot n$ where $c_{\mathsf{t}} = c_{\mathsf{t}}(\zeta, \delta)$ is constant, and,*

takes an independent and uniform random seed $Y \sim \{0,1\}^{d_{\mathsf{samp}}}$ *and outputs a random variable* Z *that is* ε*-close to a*

$$((\Delta_1, \ldots, \Delta_t), (1-\zeta)\delta)$$

block-source, where each $\Delta_{i-1} = \alpha \cdot \Delta_i$ *for* $\alpha = \frac{\zeta\delta}{4}$*. Moreover, the seed length* $d = O\Big(\frac{\log n}{\log \Delta_1} \cdot \log \frac{t}{\varepsilon}\Big)$*, and the procedure runs in time* $\widetilde{O}(n + \log^2(t/\varepsilon))$*.*

Note that for any constants $0 < \theta_1 < \theta_2 < 1$*, and any* $\varepsilon = \Omega(2^{-\sqrt{n}})$*, we can have* $\Delta_t = n^{\theta_2}$ *and* $\Delta_1 = n^{\theta_1}$ *for some* $t = O(\log n)$*, with seed length* $O(\log(1/\varepsilon))$ *and runtime* $\widetilde{O}(n)$*.*

Proof. Given our $\Delta_1, \ldots, \Delta_t$, we let $m_i = \sum_{j=1}^{i} \Delta_j$ for $j \in [t]$. Note that for $i \in [t-1]$, each $m_i = \sum_{j=1}^{i} \Delta_j \le \frac{\alpha}{1-\alpha}\Delta_{i+1}$, so in particular

$$m_t = m_{t-1} + \Delta_t \le \frac{\alpha}{1-\alpha}\Delta_t + \Delta_t \le n,$$

by choosing the constant c_{t} appropriately. Let $\Gamma\colon \{0,1\}^{d_{\mathsf{samp}}} \to [n]^{m_t}$ be the $(\gamma, \varepsilon_\Gamma)$ sampler of Lemma 3.9, set with $\varepsilon_\Gamma = \frac{1}{\log(\frac{6}{\zeta\delta})} \cdot \frac{\zeta\delta}{6} = O(1)$ and $\gamma = \frac{\varepsilon}{2t}$. Note that then,

$$d_{\mathsf{samp}} = O\left(\frac{\log n}{\log m_1} \cdot \log \frac{1}{\gamma}\right) = O\left(\frac{\log n}{\log \Delta_1} \cdot \log \frac{t}{\varepsilon}\right),$$

and indeed $m_t \le \frac{\varepsilon_\Gamma}{16} \cdot n$ can be met by, again, setting the constant c_{t} appropriately. Moreover, we have that for any $i \in [t]$,

$$W_i = \Gamma(Y)|_{[1,m_i]}$$

is a $(\gamma, \varepsilon_\Gamma)$ sampler, where $w \sim W_i$ has distinct symbols. Set $\beta = \frac{\zeta}{2}$.

Now, Lemma 2.16, instantiated with $\tau = \frac{\beta\delta}{3}$ (notice that indeed $\varepsilon_\Gamma \le \frac{\tau}{\log(1/\tau)}$), tells us that for every $i \in [t]$, denoting $A_i = X_{W_i}$, there exists a set $\mathbf{B}_i \subseteq \{0,1\}^{d_{\mathsf{samp}}}$ of bad y-s of density at most $\gamma + 2^{-\Omega(\tau n)}$, such that for any $y \notin \mathbf{B}_i$,

$$A_i | \{Y = y\} \sim \{0,1\}^{m_i}$$

has entropy rate $\delta - 3\tau \ge (1-\beta)\delta$ for every y. Letting $Z = A_t$, union-bounding over the bad y-s tells us that Z is $t \cdot (\gamma + 2^{-\Omega(n)}) \le \varepsilon$ close to some $Z' \sim \{0,1\}^{m_t}$ such that for any $i \in [t]$, $Z'_i = Z'_{[1,m_i]} \sim \{0,1\}^{m_i}$ has entropy rate $(1-\beta)\delta$.

Next, we apply the chain rule for min-entropy to argue that Z' (and hence Z) is close to a block source. To do that, we apply the chain rule for min-entropy $t-1$ times. For simplicity, abbreviate $Z^{(i)} = Z'_{[m_{i-1}+1,m_i]}$ (so note that Z'_i is the longer block, $Z'_{[1,m_i]}$, whereas $Z^{(i)}$ is its length-Δ_i suffix), so

$$Z' = \Big(Z^{(1)}, Z^{(2)}, \ldots, Z^{(t)}\Big).$$

We will argue that Z' is a block source. Applying Lemma 2.5, we know that for any $i \in [t]$,

$$\widetilde{\mathbf{H}}_\infty\Big(Z^{(i)} \mid Z^{(1)}, \ldots, Z^{(i-1)}\Big) \ge \mathbf{H}_\infty(Z^{(i)}) - \sum_{j=1}^{i-1} \Delta_j = \mathbf{H}_\infty(Z^{(i)}) - m_{i-1} \ge \mathbf{H}_\infty(Z^{(i)}) - \frac{\alpha}{1-\alpha}\Delta_i.$$

Now, $Z'_i = (Z'_{i-1}, Z^{(i)})$, so $\mathbf{H}_\infty(Z^{(i)}) \ge \mathbf{H}_\infty(Z'_i) - m_{i-1}$, and notice that

$$(1-\beta)\delta \cdot \Delta_i - m_{i-1} \ge (1-\beta)\delta \cdot \Delta_i - \frac{\alpha}{1-\alpha}\Delta_i \ge (1-\zeta)\delta \cdot \Delta_i,$$

where we used the fact that $\alpha \le \frac{(\zeta-\beta)\delta}{1-(\zeta-\beta)\delta}$.

The bound on the runtime follows easily, recalling that Γ runs in time $\widetilde{O}\big(n + \log^2(1/\gamma)\big)$. □

5.1.2 Item 3: Subsampling from the block source

To apply iterative extraction, we will our block source to have *decreasing* blocks. Here, we will use a sampler to sample from each block, using the same seed accross the blocks.

Lemma 5.3 (subsampling from a block source)**.** *There exists a deterministic procedure that given a*

$$((\Delta_1, \ldots, \Delta_t), \delta)$$

block-source $Z = (Z_1, \ldots, Z_t)$, for every $\Delta_1 \leq \ldots \leq \Delta_t$ and a constant δ, and:

- *A constant shrinkage parameter $\alpha \in (0,1)$,*
- *A constant loss parameter $\zeta \in (0,1)$,*
- *A closeness parameter $\varepsilon \in (0,1)$,*
- *An initial, maximal, block length $\ell_1 \leq \Delta_1$, and,*

takes an independent and uniform random seed $Y \sim \{0,1\}^{d_{\mathsf{samp}}}$ and outputs a random variable S that is ε-close to a $((\ell_1, \ldots, \ell_t), (1-\zeta)\delta)$ block-source, where each $\ell_{i+1} = \alpha \cdot \ell_i$, and assuming that $\ell_t \geq c_1 \log(t/\varepsilon)$ where $c_1 = c_1(\zeta, \delta)$ is a constant. Moreover, the seed length $d = \log \frac{\Delta_t}{\ell_1} + O\big(t + \log \frac{1}{\varepsilon}\big)$, and the procedure runs in time $\mathrm{polylog}(\Delta_t) \cdot \ell_1$.

Note that when $\Delta_1 = n^{\theta_1}$ and $\ell_t = n^{\beta}$ for some constants $0 < \beta < \theta_1 < 2$, $d_{\mathsf{samp}} = O(\log(n/\varepsilon))$, the procedure runs in time $O(n)$, and we can take any $\varepsilon \geq 2^{-c \cdot \ell_t}$ for some constant c that depends on ζ and δ.

Proof. For $i \in [t]$, let $m_i = \sum_{j=1}^{i} \ell_i$, recalling that $\ell_i = \alpha^{i-1}\ell_1$. For each $i \in [t]$, let $\Gamma_i \colon \{0,1\}^{d_i} \to [\Delta_i]^{\ell_i}$ be the $(\gamma, \varepsilon_\Gamma)$ be the distinct-samplers sampler of Lemma 2.21, where $\gamma = \frac{\varepsilon}{2t}$ and $\varepsilon_\Gamma = \frac{1}{\log(\frac{6}{\zeta\delta})} \cdot \frac{\zeta\delta}{6} = O(1)$. We need to make sure that each $\ell_i \geq c \cdot \frac{\log(1/\gamma)}{\varepsilon_\Gamma^2}$ for some universal constant c, and indeed that is the case, by our constraint on ℓ_t. Also, $d_i = \log(\Delta_i/\ell_i) + O(\log \frac{1}{\gamma} \cdot \mathrm{poly}(1/\varepsilon_\Gamma))$ and we set d_{samp} to be the maximal over the d_i-s, so

$$d_{\mathsf{samp}} = d_t = \log \frac{\Delta_t}{\ell_1} + t \cdot \log \frac{1}{\alpha} + O\left(\log \frac{t}{\varepsilon}\right).$$

We denote the corresponding samples by $W_i = \Gamma_i(Y|_{[1,d_i]})$, and let $S_i = Z_i|_{W_i}$. Setting $\varepsilon_i' = 2^{-(\zeta/2)\delta\Delta_i}$ and observing that $\delta\Delta_i = (1 - \frac{\zeta}{2})\delta\Delta_i + \log(1/\varepsilon_i')$, we get that Z is $\varepsilon' = \sum_i \varepsilon_i'$ close to some Z', an exact $((\Delta_1, \ldots, \Delta_t), (1-\zeta)\delta)$-source. From here onwards, assume that Z is the exact block source, and aggregate the error.

Next, we invoke Lemma 2.16 with $\tau = \frac{\zeta\delta}{6}$ (notice that indeed $\varepsilon_\Gamma \leq \frac{\tau}{\log(1/\tau)}$), and get that for every $i \in [t]$, and $z_{\mathsf{pre}} \in \{0,1\}^{\Delta_1 + \ldots + \Delta_{i-1}}$,

$$S_i \mid \{(Z_1, \ldots, Z_{i-1}) = z_{\mathsf{pre}}\}$$

is $\varepsilon_i'' = \gamma + 2^{-\Omega(\tau\Delta_i)}$-close to having min-entropy $(1 - \frac{\zeta}{2})^2\delta \cdot \ell_i \geq (1-\zeta)\delta \cdot \ell_i$. Thus, in particular, it holds if we condition on any sample from $(S_1, \ldots, S_{i-1})$, and so we have that for every $i \in [t]$,

$$(S_1, \ldots, S_{i-1}, S_i) \approx_{\varepsilon_i''} (S_1, \ldots, S_{i-1}, S_i'),$$

where S'_i has $(1-\zeta)\delta$ entropy rate. This means[17] that $(S_1, \ldots, S_t)$ has distance

$$\varepsilon' + \sum_{i=1}^{t} \varepsilon''_i \leq t \cdot (\varepsilon'_1 + \varepsilon''_1) \leq \varepsilon$$

from an $((\ell_1, \ldots, \ell_t), (1-\zeta)\delta)$ block source, where we used the fact that the $2^{-\Omega(\tau\Delta_1)}$ and $2^{-(\zeta/2)\delta\Delta_1}$ terms are at most $\frac{\varepsilon}{4t}$, which follows from the fact that $c_1 \log(t/\varepsilon) \leq \Delta_1$ for a suitable choice of c_1.

To establish the runtime, note that we simply apply Γ_i for each $i \in [t]$, which takes

$$\sum_{i=1}^{t} \ell_i \cdot \mathrm{polylog}(\Delta_i) = \mathrm{polylog}(\Delta_t) \cdot \ell_1$$

time. This concludes our lemma. □

5.1.3 Item 4: Applying a block source extractor

We now wish to extract from our decreasing-blocks block source, and for that we combine Lemmas 5.2 and 5.3 with the block source extraction of Lemma 2.23, which will give us a nearly linear-time logarithmic-seed extractor that outputs $n^{\Omega(1)}$ bits. For the Ext_i-s in Lemma 2.23, we will use the fast hash-based extractors from Lemma 2.14.

Lemma 5.4. *There exists a small constant $c > 0$ such that the following holds. For every large enough n, any constant $\delta \in (0,1)$, any $k \geq \delta n$, and any $\varepsilon \geq 2^{-n^c}$, there exists a (k, ε) extractor*

$$\mathsf{Ext}_{\mathsf{short}} \colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$$

where $d = O(\log(n/\varepsilon))$, and $m = n^c$. Moreover, given inputs $x \in \{0,1\}^n$ and $y \in \{0,1\}^d$, we can compute $\mathsf{Ext}_{\mathsf{short}}(x,y)$ in time $\widetilde{O}(n)$.

Proof. Let X be an $(n, k = \delta n)$-source. Set $\varepsilon' = \varepsilon/3$, $\theta_1 = 8/10$, $\theta_2 = 9/10$, and $\zeta = 1/10$. We first apply Lemma 5.2 with $\Delta_t = n^{\theta_2}$, $\Delta_1 = n^{\theta_1}$, and error ε', where $t = O(\log n)$ is as guaranteed from the lemma's statement. This requires a seed of length $d_1 = O(\log(1/\varepsilon')) = O(\log(1/\varepsilon))$, and in time $\widetilde{O}(n)$ we output a random variable Z_1 which is ε'-close to a $((\Delta_1, \ldots, \Delta_t), (1-\zeta)\delta)$ block source. Assume that Z_1 is exactly a block source, and aggregate the error.

Set $\beta = 7/10$, and $\gamma = 6/10 < \beta$. Set α to be the constant such that $n^\beta \cdot \alpha^{t-1} = n^\gamma$. We then apply Lemma 5.3 on Z_1 with that α, the same ζ, closeness ε' and an initial block length $\ell_1 = n^\beta$.This gives us a random variable Z_2 that is $2\varepsilon'$-close to a

$$\left((\ell_1 = n^\beta, \ldots, \ell_t = n^\gamma), \delta' \triangleq (1-\zeta)^2\delta\right)$$

block source, requires a seed of length $d_2 = O(\log(n/\varepsilon')) = O(\log(n/\varepsilon))$, and runs in time $O(n)$. Again, assume that Z_2 is exactly a block source, and aggregate the error.

For our next and final step, set $d_3 = c_{\mathsf{E}} \log(\ell_t/\varepsilon_{\mathsf{Ext}})$ where c_{E} is the constant guaranteed by Lemma 2.14. Also, let $\varepsilon_{\mathsf{Ext}} = \frac{\varepsilon'}{6t}$, and θ will be a constant whose value will be later determined. We will use the following extractors:

[17] In what follows, we use the fact that we can couple any two $X \approx_\varepsilon X'$ with $(X,Y) \approx_\varepsilon (X',Y)$, for any joint distribution (X,Y). See, e.g., [Li15, Lemma 3.20].

- Let $\mathsf{Ext}_t\colon \{0,1\}^{\ell_t}\times\{0,1\}^{d_3}\to\{0,1\}^{m_t=(1+\theta)d_3}$ be the $(k_t=(\delta'/2)\ell_t,\varepsilon_{\mathsf{Ext}})$ extractor guaranteed to us by Lemma 2.14. Notice that we need to satisfy $k_t\geq\theta d_3+c_{\mathsf{E}}\log(1/\varepsilon_{\mathsf{Ext}})$. Looking forward, we will also need that $(\delta'/2)\ell_t\leq\delta'\ell_t-\log(1/\varepsilon_{\mathsf{Ext}})$. Those constraints can be satisfied making sure that ε is at most $2^{-\Omega(\ell_t)}$, where the hidden constant depends on c_{E}.

- For each $i\in[t-1]$, let
$$\mathsf{Ext}_i\colon \{0,1\}^{\ell_i}\times\{0,1\}^{m_{i+1}}\to\{0,1\}^{m_i}$$
be the $(k_i=(\delta'/2)\ell_i,\varepsilon_{\mathsf{Ext}})$ extractor guaranteed to us by Lemma 2.14, where $m_i=(1+\theta)m_{i+1}$. We need to make sure that $m_{i+1}\geq c_{\mathsf{E}}\log(\ell_i/\varepsilon_{\mathsf{Ext}})$ and that $k_i\geq\theta m_{i+1}+c_{\mathsf{E}}\log(1/\varepsilon_{\mathsf{Ext}})$. To see that the latter holds, note that $k_i=(\delta'/2)\ell_1\cdot\alpha^{i-1}\geq n^{\gamma/2}$ and that $\theta m_{i+1}+c_{\mathsf{E}}\log(1/\varepsilon_{\mathsf{Ext}})=\theta(1+\theta)^{t-i}d_3+c_{\mathsf{E}}\log(1/\varepsilon_{\mathsf{Ext}})<n^{\gamma/2}$, if we choose θ to be a small enough constant (with respect to the constant $\frac{\log n}{t}$) and ε to be, again, at most $2^{-\Omega(\ell_t)}$. Also, here too, record that $(\delta'/2)\ell_i\leq\delta'\ell_i-\log(1/\varepsilon_{\mathsf{Ext}})$, which follows easily, since the ℓ_i-s increase.

Everything is in place to apply our block source extraction, Lemma 2.23, on Z_2 and an independent and uniform seed of length d_3.[18] We get that BSExt outputs Z_3 of length $m_1=n^{\Omega(1)}$, which is $2t\varepsilon_{\mathsf{Ext}}\leq\varepsilon'$ close to uniform, and runs in time $O\left(\sum_{i=1}^{t}\ell_i\log\ell_i\right)=O(n)$.

To conclude, note that the overall error of our extractor is at most $3\varepsilon'=\varepsilon$, and the seed length is $d_1+d_2+d_3=O(\log(n/\varepsilon))$. □

5.1.4 Improving the output length

The extractor $\mathsf{Ext}_{\mathsf{short}}$ from Lemma 5.4 only outputs $n^{\Omega(1)}$ bits. Here, we will use an extractor $\mathsf{Ext}_{\mathsf{aux}}$ that outputs a linear fraction of the entropy but requires a (relatively) long seed, and use Lemma 2.26 to boost the output length. For $\mathsf{Ext}_{\mathsf{aux}}$, we will again use a sample-then-extract extractor, however this time, we can use *independent* samples to create a block source with exponentially decreasing blocks. This setting is easier, and we can simply use the original [NZ96] construction. Since a similar construction will be analyzed later in the paper (including a time complexity analysis), we choose to employ it instead of revisiting [NZ96].

Corollary 5.5. *There exist constants $\tau,c\in(0,1)$ and $C>1$, such that for every positive integer n, and any $\varepsilon\geq 2^{-n^c}$, there exists a strong $(k=(1-\tau)n,\varepsilon)$ extractor*
$$\mathsf{Ext}_{\mathsf{out}}\colon \{0,1\}^n\times\{0,1\}^d\to\{0,1\}^m$$
where $d=O(\log n\cdot\log(n/\varepsilon))$, and $m=ck-C\log(1/\varepsilon)$. Moreover, given inputs $x\in\{0,1\}^n$ and $y\in\{0,1\}^d$, we can compute $\mathsf{Ext}_{\mathsf{out}}(x,y)$ in time $\widetilde{O}(n)$.

The correctness follows from Lemma 5.9 (without the need for a preliminary condensing step), employed with the hash functions of Lemma 2.14.

Plugging-in $\mathsf{Ext}_{\mathsf{out}}$ and $\mathsf{Ext}_{\mathsf{short}}$ into Lemma 2.26 readily gives the following result.

Lemma 5.6. *There exist constants $\tau,c\in(0,1)$ such that for every positive integer n, and any $\varepsilon\geq 2^{-n^c}$, there exists a $(k=(1-\tau)n,\varepsilon)$ extractor $\mathsf{Ext}\colon\{0,1\}^n\times\{0,1\}^d\to\{0,1\}^m$ where $d=O(\log(n/\varepsilon))$, and $m=ck$. Moreover, given inputs $x\in\{0,1\}^n$ and $y\in\{0,1\}^d$, we can compute $\mathsf{Ext}(x,y)$ in time $\widetilde{O}(n)$.*

[18] Note that here, we use Lemma 2.23 with $n_i=(1+\theta)^{-i}n_1$ and $k_i\geq\theta n_i-\log(1/\varepsilon_i)$. The slack in entropy is needed since we work with the notion of block sources that also allows average conditional min-entropy. We can thus use the fact that under such setting of parameters, every extractor is an average case extractor with only a slight loss in parameters (see, e.g., [DORS08]). We omit the easy proof.

To boost the output length from $\Omega(k)$ to $(1-\eta)k$ for any constant $\eta > 0$, we apply Lemma 2.22 a constant number of times depending only on η (that is, we simply apply Ext with independent seeds and concatenate the outputs). To go from any min-entropy k to entropy rate $1-\tau$, we first apply a condenser, either the one from Theorem 3.10 or the one from Theorem 3.11. Specifically, when $k \geq C \log^2(n/\varepsilon)$, we can use Theorem 3.10 which takes $\widetilde{O}(n)$ time. When k is smaller, we can use Theorem 3.11, but this requires an extra preprocessing time which takes $T_{\mathsf{pre}} = \mathrm{polylog}(1/\varepsilon)$ times. Note that the bound on ε from Lemma 5.6 translates to $\varepsilon \geq 2^{-k^c}$, so we can (if needed) modify c so that $T_{\mathsf{pre}} = O(n)$. This finally gives us our main theorem for this section, Theorem 5.1, apart from the strongness property, which we now discuss.

The non-recursive construction is strong. In what follows, we refer to the itemized list in the beginning of the section. The condensing step, Item 1, is strong, since we use strong condensers. Next, inspecting the proofs of Lemmas 5.2 and 5.3, we see that both samplings procedures yield a good sample with high probability over the fixing of the seed, so Items 2 and 3 hold in a strong manner as well. Item 4 follows by applying a block source extractor, which is strong since the extraction steps output the seed. Thus, the extractor $\mathsf{Ext}_{\mathsf{short}}$ from Lemma 5.4 is in fact strong. For the output-extending phase, Lemma 2.26 readily tells us that the extractor from Lemma 5.6 is strong. Finally, we apply that extractor several times with independent seeds, and the strongness of that procedure is guaranteed from Lemma 2.22.

5.2 A Recursive Construction

In this section, we prove the following.

Theorem 5.7 (recursive construction)**.** *For any constant $\eta > 0$ there exists a constant $C > 0$ such that the following holds. For any positive integers n, $k \leq n$, and any $\varepsilon > 0$ such that $k \geq C \log(n/\varepsilon)$ there exists a strong (k, ε)-seeded extractor*

$$\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$$

with seed length $d \leq C \log(n/\varepsilon)$ and output length $m \geq (1-\eta)k$. Furthermore,

- *if $k \geq 2^{C \log^* n} \cdot \log^2(n/\varepsilon)$, then Ext is computable in time $\widetilde{O}(n)$;*
- *if $k < 2^{C \log^* n} \cdot \log^2(n/\varepsilon)$, then Ext is computable in time $\widetilde{O}(n)$ after a preprocessing step that corresponds to finding primitive elements of $O(\log \log n)$ fields $\mathbb{F}_q$ with orders $q \leq \mathrm{poly}(n/\varepsilon)$ powers of 2.*

In a nutshell, our construction behind Theorem 5.7 works by considering two cases. If $\varepsilon > Cn^3 \cdot 2^{-k/\log k}$, then we instantiate the recursive approach of Srinivasan and Zuckerman [SZ99] appropriately. Otherwise, we apply the recursive approach of Guruswami, Umans, and Vadhan [GUV09].

5.2.1 The (extremely) low-error case

In this section, we consider the lower error case of Theorem 5.7 where $\varepsilon \leq Cn^3 \cdot 2^{-k/\log k}$. We instantiate the recursive approach from [GUV09, Section 4.3.3] appropriately, and analyze its time complexity. Crucially, because of our upper bound on ε, we will only need to run $O(\log \log n)$ levels of their recursive approach.

In order to obtain the statement of Theorem 5.7 for output length $m \geq (1-\eta)k$ with η an arbitrarily small constant, it suffices to achieve output length $m = \Omega(k)$ and then apply Lemma 2.22 a constant number of times depending only on η. Therefore, we focus on achieving output length $m = \Omega(k)$.

Theorem 5.8. *There exist constants $c, C > 0$ such that the following holds. For any positive integers n and $k \le n$ and any $\varepsilon \in (0, Cn^3 \cdot 2^{-k/\log k}]$ further satisfying $k > C\log(n/\varepsilon)$, there exists a strong (k,ε)-seeded extractor $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with seed length $d \le C\log(n/\varepsilon)$ and output length $m \ge k/3$.*

Furthermore, Ext is computable in time $\widetilde{O}(n)$ after a preprocessing step that corresponds to finding primitive elements of $O(\log\log n)$ fields $\mathbb{F}_q$ with orders $q \le \mathrm{poly}(n/\varepsilon)$, each a power of 2.

Proof. We discuss our instantiation of the recursive approach from [GUV09] in detail, as it will be relevant to the time complexity analysis. Let $\varepsilon_0 = \varepsilon/\log^C n$ and $d = C\log(n/\varepsilon_0) = O(\log(n/\varepsilon))$ for a large enough constant $C > 0$ to be determined later. For an integer $k \ge 0$, let $i(k) = \left\lceil \log\left(\frac{k}{8d}\right)\right\rceil$, which determines the number of levels in our recursion. It will be important for bounding the time complexity of this construction to observe that

$$i(k) = O(\log\log n) \tag{2}$$

because $\varepsilon \le Cn^3 \cdot 2^{-k/\log k}$. For each k, we define a family of strong $(k, \varepsilon_{i(k)})$-seeded extractors $\mathsf{Ext}_{i(k)}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with $\varepsilon_{i(k)} \le 9\varepsilon_{i(k/3)} + 63\varepsilon_0$ when $i(k) > 0$ by induction on $i(k)$. Solving this recursion yields $\varepsilon_{i(k)} = 2^{O(i(k))} \cdot \varepsilon_0 \le \varepsilon$, provided that $\varepsilon_0 = \varepsilon/\log^C n$ for a sufficiently large constant $C > 0$.

Base case. For the base case $i(k) = 0$, which holds when $k \le 8d$, we choose Ext_0 to be the (k,ε_0)-seeded extractor defined as follows. On input an (n,k)-source X,

1. Apply the lossy RS strong condenser RSCond (Theorem 3.11) on X, instantiated with $\alpha = 1/400$ and error $\varepsilon_0' = \varepsilon_0/2$. When C is large enough we have $k \ge (2+\alpha)\log(1/\varepsilon_0')$, and require a seed Y_1 of length $d_1 \le C_0\log(n/\varepsilon_0')$, for some constant $C_0 > 0$. The corresponding output X' satisfies $Y_1 \circ X' \approx_{\varepsilon_0'} Y_1 \circ Z$, for some (n', k')-source Z with $k' \ge (1-2\alpha)n' = (1-1/200)n'$.

2. Let $\mathsf{Ext}_0'\colon \{0,1\}^{n'} \times \{0,1\}^{d_2} \to \{0,1\}^{m'}$ be the average-case strong (k', ε_0')-seeded extractor from Lemma 2.25 instantiated with $t = 10$, which requires a seed Y_2 of length $d_2 \le k'/10 + C_0'\log(n'/\varepsilon_0')$ for some constant $C_0' > 0$ and has output length $m' \ge k'/2$. The conditions for the invocation of Lemma 2.25 with $t = 10$ are satisfied since $k' \ge (1-1/200)n' = (1 - \frac{1}{20t})n'$ and
$$2^{-n'/500} \le 2^{-k/500} \le (\varepsilon_0/n)^{C/500} \le \varepsilon_0',$$
where the second inequality uses the theorem's hypothesis that $k \ge C\log(n/\varepsilon)$ with $C > 0$ a sufficiently large constant.

We set $Y = Y_1 \circ Y_2$ and define $\mathsf{Ext}_0(X, Y) = \mathsf{Ext}_0'(\mathsf{RSCond}(X, Y_1), Y_2)$. From the discussion above, we have

$$Y \circ \mathsf{Ext}_0(X,Y) = Y_1 \circ Y_2 \circ \mathsf{Ext}_0'(\mathsf{RSCond}(X,Y_1),Y_2) \approx_{\varepsilon_0'} Y_1 \circ Y_2 \circ \mathsf{Ext}_0'(Z,Y_2) \approx_{\varepsilon_0'} Y_1 \circ Y_2 \circ U_{m'}.$$

Therefore, the triangle inequality implies that Ext_0 is an average-case strong $(k, 2\varepsilon_0' = \varepsilon_0)$-seeded extractor. It remains to argue about the seed length, output length, and time complexity of Ext_0. The seed length of Ext_0 is

$$d_1 + d_2 \le k'/10 + (C_0 + C_0')\log(n'/\varepsilon_0') \le 0.8d + (C_0 + C_0')\log(n'/\varepsilon_0') \le d,$$

provided that $d = C\log(n/\varepsilon)$ with C a sufficiently large constant. The output length of Ext_0 is $m' \ge k'/2 \ge k/3$, since $k' \ge (1-1/200)k$. Finally, both steps above take time $\widetilde{O}(n)$, and so Ext_0 can be computed in time $\widetilde{O}(n)$ after a $\mathrm{polylog}(1/\varepsilon)$ preprocessing step.

Induction step. When $i(k) > 0$, we assume the existence of the desired average-case strong extractors $\mathsf{Ext}_{i(k')}$ for all $i(k') < i(k)$ as the induction hypothesis. More precisely, we assume that for all k' such that $i(k') < i(k)$ there exists a family of average-case strong $(k', \varepsilon_{i(k')})$-seeded extractors $\mathsf{Ext}_{i(k')} \colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{k'/3}$ parameterized by n computable in time $\widetilde{O}(n)$ after a one-time preprocessing step. We proceed as follows on input an (n,k)-source X:

1. Apply the lossy RS strong $(k, k', \varepsilon_1 = \varepsilon_0^2)$-condenser RSCond (Theorem 3.11) on X with $\alpha = 1/20$ and a seed Y_{RS} of length $d_{\mathsf{RS}} \le C_{\mathsf{RS}} \log(n/\varepsilon_0)$. Since $k > 8d \ge \frac{\log(1/\varepsilon_1)}{\alpha(1+2\alpha)}$ if C is a large enough constant, by the second part of Theorem 3.11 we know that with probability at least $1 - \varepsilon_0$ over the choice of $Y_{\mathsf{RS}} = y$ it holds that the corresponding condenser output X' is ε_0-close to some (n', k')-source Z with $k' \ge (1-2\alpha)n' = 0.9n'$. For the sake of exposition, from here onwards we work under such a good choice of the seed Y_{RS}, and we will add the ε_0 slack term to the final error.

2. Split $X' = X'_1 \circ X'_2$ with $|X'_1| = |X'_2| = n'/2 \triangleq n''$. By Lemma 2.24 instantiated with n' and $\Delta = 0.1n'$ and the fact that X' is ε_0-close to an (n', k')-source, we get that $X'_1 \circ X'_2$ is $(\varepsilon_{\mathsf{RS}} + 2\varepsilon_0 = 3\varepsilon_0)$-close to an $((n'', n''), k''/n'')$-block-source $W_1 \circ W_2$ with

$$k'' \ge k'/2 - \Delta - \log(1/\varepsilon_0) \ge 0.4n' - \log(1/\varepsilon_0) \ge k/3, \tag{3}$$

since $n' \ge k > d = C \log(n/\varepsilon_0)$ for a sufficiently large constant $C > 0$.

3. Apply the lossy RS strong $(k'', k''', \varepsilon_1 = \varepsilon_0^2)$-condenser RSCond' (Theorem 3.11) to X'_2 with $\alpha = 1/800$ and a seed Y'_{RS} of length at most $d'_{\mathsf{RS}} = C'_{\mathsf{RS}} \log(n''/\varepsilon_1) \le C'_{\mathsf{RS}} \log(n/\varepsilon_0)$. From Item 2 and the data-processing inequality, we know that

$$Y'_{\mathsf{RS}} \circ X'_1 \circ X''_2 = Y'_{\mathsf{RS}} \circ X'_1 \circ \mathsf{RSCond}(X'_2, Y'_{\mathsf{RS}}) \approx_{3\varepsilon_0} Y'_{\mathsf{RS}} \circ W_1 \circ \mathsf{RSCond}(W_2, Y'_{\mathsf{RS}}). \tag{4}$$

Since $(W_2 | W_1 = w_1)$ is a k''-source for any w_1 in the support of W_1, we conclude from Theorem 3.11 and Equation (4) that

$$Y'_{\mathsf{RS}} \circ W_1 \circ \mathsf{RSCond}(W_2, Y'_{\mathsf{RS}}) \approx_{\varepsilon_1} Y'_{\mathsf{RS}} \circ W_1 \circ \widetilde{W_2},$$

where $\widetilde{W_2} \sim \{0,1\}^{n'''}$ and $\mathbf{H}_\infty(Y'_{\mathsf{RS}} \circ \widetilde{W_2} | W_1 = w_1) \ge k''' + d'_{\mathsf{RS}}$ for all w_1 in the support of W_1, with $n''' \ge k'' \ge k''' \ge (1 - 1/400)n'''$. This is a valid invocation since $k'' \ge k/3 > 8d/3 > d \ge \frac{\log(1/\varepsilon_1)}{\alpha(1+2\alpha)}$ by Equation (3). Therefore, by the second part of Theorem 3.11, with probability at least $1 - \varepsilon_0$ over the choice of of $Y'_{\mathsf{RS}} = y'$ we get that

$$(W_1 \circ \widetilde{W_2} | Y'_{\mathsf{RS}} = y') \approx_{\varepsilon_0} W_1 \circ W'_2, \tag{5}$$

where $W'_2 \sim \{0,1\}^{n'''}$ satisfies $\mathbf{H}_\infty(W'_2 | W_1 = w_1) \ge k''' \ge (1 - 1/400)n'''$. Fix such a good fixing of Y'_{RS} from now onwards. As before, we will account for the probability ε_0 of fixing a bad seed in the final extractor error. Then, by combining Equations (4) and (5) we get that $X'_1 \circ X''_2$ is $(\varepsilon_{\mathsf{BS}} = 4\varepsilon_0)$-close to an $((n'', n'''), k'', k''')$-block source.

4. We will now apply block source extraction to $X'_1 \circ X''_2$, which we recall is $(\varepsilon_{\mathsf{BS}} = 4\varepsilon_0)$-close to an $((n'', n'''), k'', k''')$-block source. We instantiate Lemma 2.23 with Ext_2 being the strong extractor from Lemma 2.25 with source input length n''', min-entropy requirement k''', error $\varepsilon_{\mathsf{BExt}} = \varepsilon_0$, output length d, and $t = 16$. This requires a seed of length $d_{\mathsf{BExt}} \le d/16 +$

$C_0' \log(n/\varepsilon_0)$. This instantiation of Lemma 2.25 is valid since $k''' \geq (1-1/400)n''' > (1-\frac{1}{20t})n'''$ and

$$k''' \geq 0.95n''' \geq 0.95k'' \geq \frac{0.95k}{3} > \frac{0.95 \cdot 8d}{3} > 2d,$$

where we used the fact that $i(k) > 0$, and so $k > 8d$. For Ext_1 we choose the average-case strong extractor $\mathsf{Ext}_{i(k/3)}$ (recall that $k'' \geq k/3$ and note that $i(k/3) < i(k)$) with input length n'', entropy requirement $k/3$, error $\varepsilon_{i(k/3)}$, output length at least $(k/3)/3 = k/9$, and seed length d guaranteed by the induction hypothesis above.

Items 1 to 4 above yield a strong seeded extractor $\mathsf{Ext}'_{i(k)} \colon \{0,1\}^n \times \{0,1\}^{d'} \to \{0,1\}^{m'}$ with min-entropy requirement k, error $\varepsilon' = \varepsilon_{i(k/3)} + \varepsilon_{\mathsf{BExt}} + \varepsilon_{\mathsf{BS}} + 2\varepsilon_0 = \varepsilon_{i(k/3)} + 7\varepsilon_0$ (where the $2\varepsilon_0$ term comes from the two fixings of the seeds in the two condensing steps in Items 1 and 3), seed length

$$d' = d_{\mathsf{BExt}} + d'_{\mathsf{RS}} + d_{\mathsf{RS}} \leq d/16 + C' \log(n/\varepsilon_0),$$

for some constant $C' > 0$, and output length $m' = k/9$.

To conclude the definition of $\mathsf{Ext}_{i(k)}$, we need to increase the output length of $\mathsf{Ext}'_{i(k)}$ from $k/9$ to $k/3$. To that end, we use Lemma 2.22. Applying Lemma 2.22 once with $\mathsf{Ext}_1 = \mathsf{Ext}'_{i(k_1)}$ with $k_1 = k$ and $\mathsf{Ext}_2 = \mathsf{Ext}'_{i(k_2)}$ with $k_2 = k - k/9 - 1 = 8k/9 - 1$ and $g = 1$ yields a strong $(k, 3\varepsilon')$-seeded extractor $\mathsf{Ext}''_{i(k)}$ with output length $(k_1 + k_2)/9 \geq k(1 - (8/9)^2) - 1$ and seed length $2(d/16 + C' \log(n/\varepsilon_0)) = d/8 + 2C' \log(n/\varepsilon_0)$. Applying Lemma 2.22 again with $\mathsf{Ext}_1 = \mathsf{Ext}''_{i(k_1)}$ for $k_1 = k$ and $\mathsf{Ext}_2 = \mathsf{Ext}''_{i(k_2)}$ for $k_2 = (8/9)^2 k$ and $g = 1$ yields a strong $(k, 9\varepsilon')$-seeded extractor with output length $m \geq k(1 - (8/9)^4) - 1 \geq k/3$ and seed length $2(d/8 + 2C' \log(n/\varepsilon_0)) = d/4 + 4C' \log(n/\varepsilon_0) \leq d$, which we set as $\mathsf{Ext}_{i(k)}$. This second invocation of Lemma 2.22 is also valid, since $k_2 = (8/9)^2 k = k - (k(1 - (8/9)^2) - 1) - 1 = k_1 - m_1 - g$. Note that the error $\varepsilon_{i(k)} = 9\varepsilon' = 9\varepsilon_{i(k/3)} + 63\varepsilon_0$, as desired.

Time complexity and final error. It remains to analyze the time complexity and the overall error of the recursive procedure above. Evaluating $\mathsf{Ext}_{i(k)}$ requires at most eight evaluations of the condenser from Theorem 3.11, four evaluations of the fast hash-based extractor from Lemma 2.25, four evaluations of $\mathsf{Ext}_{i(k'')}$ for some $i(k'') < i(k)$, and simple operations that can be done in time $\widetilde{O}(n)$. This means that the overall time complexity is $4^{i(k)} \cdot \widetilde{O}(n) = \widetilde{O}(n)$ after a one-time preprocessing step independent of the source and seed, since $4^{i(k)} = \mathrm{poly}(\log n)$ by Equation (2). This preprocessing step corresponds to finding primitive elements for $O(\log \log n)$ fields $\mathbb{F}_q$ with orders $q \leq \mathrm{poly}(n/\varepsilon_0) = \mathrm{poly}(n/\varepsilon)$ powers of 2. Furthermore, $\varepsilon_{i(k)} = O(\varepsilon_0 + \varepsilon_{i(k/3)})$ for all k, and so $\varepsilon_{i(k)} = 2^{O(i(k))}\varepsilon_0 = \mathrm{poly}(\log n) \cdot \varepsilon_0 \leq \varepsilon$ provided that $\varepsilon_0 \leq \varepsilon / \log^C n$ for a large enough constant $C > 0$. □

5.2.2 The (relatively) high-error case

In this section, we consider the higher error case where $\varepsilon \geq Cn^3 \cdot 2^{-k/\log k}$. We instantiate the recursive approach of Srinivasan and Zuckerman [SZ99, Section 5.5] appropriately with our building blocks and analyze its complexity.

Lemma 5.9 (analogous to [SZ99, Corollary 5.10], with different instantiation and additional complexity claim)**.** *There exist constants $c, C > 0$ such that the following holds. Suppose that for any positive integers n_0, $k_0 = 0.7n_0$, and some $\varepsilon_0 = \varepsilon_0(n_0) \geq 2^{-ck_0}$ and $m_0 = m_0(n_0)$ there exists a strong (k_0, ε_0)-seeded extractor $\mathsf{Ext}_0 \colon \{0,1\}^{n_0} \times \{0,1\}^{d_0} \to \{0,1\}^{m_0}$ with seed length $d_0 \leq u \cdot \log(n_0/\varepsilon_0) \leq k_0$.*

Then, for any positive integers n and $k \le n$ there exists a family of strong (k,ε)-seeded extractors $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with error $\varepsilon \le C \log u \cdot \varepsilon_0(ck)$, seed length $d \le C \log u \cdot \log(n/\varepsilon_0(ck))$, and output length $m \ge m_0(ck)$. Furthermore,

1. *If Ext_0 is computable in time $\widetilde{O}(n_0)$ and $k \ge C \log^2(n/\varepsilon)$, then Ext is computable in time $\widetilde{O}(n)$;*
2. *If Ext_0 is computable in time $\widetilde{O}(n_0)$ after a preprocessing step corresponding to finding primitive elements of j fields $\mathbb{F}_q$ of orders $q \le \mathrm{poly}(n/\varepsilon_0)$, then Ext is computable in time $\widetilde{O}(n)$ after a preprocessing step corresponding to finding primitive elements of $j+1$ fields $\mathbb{F}_q$ of orders $q \le \mathrm{poly}(n/\varepsilon_0)$.*

Proof. We begin by setting up relevant notation:

- Let $C_{\mathsf{blocks}} \ge 1$ be a constant to be determined. Set $\ell_0 = \frac{k}{100 \cdot C_{\mathsf{blocks}}}$ and $k_0 = 0.7\ell_0$. For $\varepsilon_0 = \varepsilon_0(\ell_0)$ and $m_0 = m_0(\ell_0)$, we define $\ell_1 = C_{\mathsf{blocks}} \cdot u \log(\ell_0/\varepsilon_0)$. Then, we define $\ell_i = 0.9\ell_{i-1}$ for all $i \ge 2$. The ℓ_i's will be block lengths for a block source Z. In particular, when performing block source extraction from Z we will instantiate Ext_0 with input length $n_0 = \ell_0$.
- Define $m_1 = u \cdot \log(\ell_0/\varepsilon_0)$ and $m_i = 0.9m_{i-1}$ for all $i \ge 2$. The m_i's will be output lengths for block source extraction from Z.
- Set $t = 1 + \frac{\log(u/\log u)}{\log(1/0.9)}$. This will be the number of blocks of Z. We have $m_t = 0.9^{t-1}m_1 = \log u \cdot \log(\ell_0/\varepsilon_0)$. Furthermore, since $\ell_1 = C_{\mathsf{blocks}} \cdot m_1$, we also have that $\ell_i = C_{\mathsf{blocks}} \cdot m_i$ for all $i \ge 1$.

Let X be an arbitrary (n,k)-source. The extractor $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ proceeds as follows on input X:

1. Using a fresh seed Y_{Cond} of length $C_{\mathsf{Cond}} \log(n/\varepsilon_0)$, apply a strong (k, k', ε_0^2)-condenser Cond to X. If $k \ge C \log^2(n/\varepsilon_0)$ for an apropriately large constant $C > 0$, then we instantiate Cond with the lossless KT $(k, k' = k, \varepsilon_{\mathsf{Cond}})$-condenser (Theorem 3.10). Otherwise, we instantiate Cond with the lossy RS $(k, k' \ge 0.95k, \varepsilon_{\mathsf{Cond}})$-condenser (Theorem 3.11) instantiated with $\alpha = 0.05$. By the second part of either Theorem 3.10 or Theorem 3.11, we get that with probability at least $1 - \varepsilon_0$ over the choice of $Y_{\mathsf{Cond}} = y$ it holds that $X' = \mathsf{Cond}(X, y)$ is ε_0-close to an (n', k')-source with $k' \ge 0.95n'$. From here onwards we work under such a good fixing $Y_{\mathsf{Cond}} = y$, and will account for the ε_0 error term in the final extractor error later on.
2. We use X' and Lemma 2.16 to generate a block source Z with geometrically decreasing block lengths $\ell_0, \ell_1, \ldots, \ell_t$ defined above.

   For each $i = 0, 1, \ldots, t$, let $\mathsf{Samp}_i\colon \{0,1\}^{r_i} \to [n']^{\ell_i}$ be the $(\theta = 1/100, \gamma = \varepsilon_0)$-averaging sampler from Lemma 2.21 with input length $r_i = C_{\mathsf{Samp}} \log(n'/\varepsilon_0)$ for some constant $C_{\mathsf{Samp}} > 0$. We choose the constant C_{blocks} above to be large enough so that $n' \ge \ell_i \ge \ell_t \ge C'_{\mathsf{Samp}} \log(1/\varepsilon_0)/\theta^2$ for all $i \in [t]$, where C'_{Samp} is the constant C from Lemma 2.21. To see that $\ell_i \le n'$ for $i = 0, \ldots, t$ (and so indeed Lemma 2.21 can be applied to obtain ℓ_i samples), note that

$$\sum_{i=0}^{t} \ell_i \le \sum_{i=0}^{\infty} \ell_i = 10\ell_1 + \ell_0 \le k/9 < n'. \tag{6}$$

   The second-to-last inequality uses the fact that

$$\ell_1 = C_{\mathsf{blocks}} \cdot u \log(\ell_0/\varepsilon_0) \le C_{\mathsf{blocks}} \cdot k_0 \le C_{\mathsf{blocks}} \cdot \ell_0 = k/100,$$

where the first inequality holds since $u \log(\ell_0/\varepsilon_0) \le k_0$ is an hypothesis in the lemma statement. We also assume that $\varepsilon_0 \ge 2^{-ck_0}$ for a constant $c > 0$ small enough so that

$$\ell_0 = \frac{k}{100 C_{\mathsf{blocks}}} \ge C'_{\mathsf{Samp}} \cdot ck_0/\theta^2 \ge C'_{\mathsf{Samp}} \log(1/\varepsilon_0)/\theta^2,$$

where we recall that $k_0 = 0.7\ell_0$, meaning that the conditions of Lemma 2.21 are satisfied for all $i = 0, \dots, t$.

For each $i = 0, 1, \dots, t$, let Y_i be a fresh seed of length r_i. We set the i-th block as $Z_i = X'_{\mathsf{Samp}(Y_i)}$. By Lemma 2.16 instantiated with X' and Samp_0, we conclude that

$$Y_0 \circ Z_0 \approx_{\varepsilon_0 + 2^{-c_{\mathsf{Samp}} k'}} Y_0 \circ Z'_0,$$

with $c_{\mathsf{Samp}} > 0$ an absolute constant guaranteed by Lemma 2.16, where $(Z'_0 | Y_0 = y)$ is an $(\ell_0, 0.9\ell_0)$-source for every y. We now argue how this guarantee extends to more blocks. Consider an arbitrary i and fixings $Y_0 = y_0, \dots, Y_{i-1} = y_{i-1}$. Then, Lemma 2.6 with $\delta = 2^{-c_{\mathsf{Samp}} k}$ and $\ell = k/9$ (from the upper bound in Equation (6)) implies that

$$\mathbf{H}_\infty(X | (Z'_0 | Y_0 = y_0) = z_0, \dots, (Z'_{i-1} | Y_{i-1} = y_{i-1}) = z_{i-1}) \ge 0.8n'$$

except with probability at most $2^{-c_{\mathsf{Samp}} k}$ over the choice of $z_0, \dots, z_{i-1}$, which we can absorb into the statistical distance, since $k' \ge 0.95n' \ge 0.95k$. Consequently, from Lemma 2.16 we get that

$$Y_0, Z'_0, \dots, Y_{i-1}, Z'_{i-1}, Y_i, Z_i = X_{\mathsf{Samp}(Y_i)} \approx_{\varepsilon_0 + 2 \cdot 2^{-c_{\mathsf{Samp}} k}} Y_0, Z'_0, \dots, Y_{i-1}, Z'_{i-1}, Y_i, Z'_i, \tag{7}$$

where $(Z'_i | Y_0 = y_0, Z'_0 = z_0, \dots, Y_{i-1} = y_{i-1}, Z'_{i-1} = z_{i-1}, Y_i = y_i)$ is an $(\ell_i, 0.7\ell_i)$-source for any choice of $y_0, z_0, \dots, y_{i-1}, z_{i-1}, y_i$. Combining Equation (7) with the triangle inequality over all $0 \le i \le t$, we conclude that $Z = Z_0 \circ Z_1 \circ \cdots \circ Z_t$ is $\varepsilon_{\mathsf{block}}$-close to an exact $(\ell_0, \dots, \ell_t, 0.7)$-block-source Z', where $\varepsilon_{\mathsf{block}} = (t+1)(\varepsilon_0 + 2 \cdot 2^{-c_{\mathsf{Samp}} k})$.

3. We apply block source extraction (Lemma 2.23) to $Z = Z_0 \circ Z_1 \circ \cdots \circ Z_t$. More precisely, let $\mathsf{BExt} \colon \{0,1\}^{\ell_0} \times \cdots \times \{0,1\}^{\ell_t} \times \{0,1\}^{d_t} \to \{0,1\}^{m_0}$ be the strong $(k_0, k_1, \dots, k_t, (t+1)\varepsilon_0)$-block-source extractor with $k_i = 0.7\ell_i$ obtained via Lemma 2.23 as follows. We instantiate Ext_0 with the strong extractor promised by the lemma statement with seed length $d_0 \le u \cdot \log(\ell_0/\varepsilon_0) = m_1$. For $i \in [t]$, we instantiate $\mathsf{Ext}_i \colon \{0,1\}^{\ell_i} \times \{0,1\}^{d_i} \to \{0,1\}^{m_i}$ as the strong $(k_i = 0.7\ell_i, \varepsilon_0)$-seeded extractor from Lemma 2.14 with seed length $d_i = 2m_i + 4\log(\ell_i/\varepsilon_0) + 8$. We choose the constant C_{blocks} to be large enough so that

$$m_i = \ell_i / C_{\mathsf{blocks}} \le 0.7\ell_i - 16\log(4/\varepsilon_0) = k_i - 16\log(4/\varepsilon_0),$$

as required by Lemma 2.14. This is possible since by choosing C_{blocks} large enough we have

$$\ell_i \ge \ell_t = C_{\mathsf{blocks}} \cdot m_t = C_{\mathsf{blocks}} \log u \cdot \log(\ell_0/\varepsilon_0) \ge 100 \log(4/\varepsilon_0)$$

for all $i \in [t]$, and so $0.7\ell_i - 16\log(4/\varepsilon_0) \ge \ell_i/2$ for all $i \in [t]$. Furthermore, for any $i \ge 2$ the output length m_i of Ext_i satisfies

$$d_i + m_i = 3m_i + 4\log(n/\varepsilon_0) + 8 \ge 2m_{i-1} + 4\log(n/\varepsilon_0) + 8 \ge d_{i-1},$$

where we recall that $m_i = m_{i-1}/0.9$ for $i \ge 2$. Finally, the output length of Ext_1 satisfies $d_1 + m_1 \ge m_1 \ge d_0$, where we recall that d_0 is the seed length of Ext_0.

Let Y_{BExt} be a fresh seed of length d_t. With the desired upper bound on the seed length d from the lemma's statement in mind, we note that

$$d_t \le 2m_t + 4\log(\ell_t/\varepsilon_0) + 8 \le 2\log u \cdot \log(\ell_0/\varepsilon_0) + 4\log(\ell_0/\varepsilon_0) \le 6\log u \cdot \log(n/\varepsilon_0), \tag{8}$$

since $\ell_0 \le k \le n$. By Lemma 2.23, we get that

$$Y_{\mathsf{BExt}} \circ \mathsf{BExt}(Z, Y_{\mathsf{BExt}}) \approx_{\varepsilon_{\mathsf{block}}} Y_{\mathsf{BExt}} \circ \mathsf{BExt}(Z', Y_{\mathsf{BExt}}) \approx_{(t+1)\varepsilon_0} U_{d_t+m_0}.$$

Applying the triangle inequality, we conclude that

$$Y_{\mathsf{BExt}} \circ \mathsf{BExt}(Z, Y_{\mathsf{BExt}}) \approx_{\varepsilon_{\mathsf{block}}+(t+1)\varepsilon_0} U_{d_t+m_0}.$$

We now define our final strong extractor $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{m_0}$ (recall that we abbreviate $m_0 = m_0(\ell_0)$). Choose our overall seed to be $Y = Y_{\mathsf{Cond}} \circ Y_0 \circ \cdots \circ Y_t \circ Y_{\mathsf{BExt}}$ and set $\mathsf{Ext}(X, Y) = \mathsf{BExt}(Z, Y_{\mathsf{BExt}})$. By the discussion above, Ext is a strong (k, ε)-extractor with error (recall that we abbreviate $\varepsilon_0 = \varepsilon_0(\ell_0)$)

$$\varepsilon = 2\varepsilon_0 + \varepsilon_{\mathsf{block}} + (t+1)\varepsilon_0 \le (2t+4)(\varepsilon_0 + 2 \cdot 2^{-c_{\mathsf{Samp}}k})$$

for a sufficiently large constant C since $t = O(\log u)$ (where one of the ε_0 terms comes from fixing the seed in the condensing step of Item 1), and seed length

$$d = |Y_{\mathsf{Cond}}| + |Y_{\mathsf{BExt}}| + \sum_{i=0}^{t} |Y_i| \le C_{\mathsf{Cond}} \log(n/\varepsilon_0) + d_t + (t+1)C_{\mathsf{Samp}} \log(n'/\varepsilon_0) \le C \log u \cdot \log(n/\varepsilon_0)$$

provided that C is large enough (again since $t = O(\log u)$), as desired. We used Equation (8) to bound d_t and obtain the last inequality.

Time complexity. It remains to analyze the time complexity of Ext. If $k \ge C\log^2(n/\varepsilon_0)$ with C a sufficiently large constant, then Item 1 takes time $\widetilde{O}(n)$. Item 2 takes time $t \cdot \widetilde{O}(n) = \widetilde{O}(n)$, since $t = O(\log u) = O(\log n)$ and each averaging sampler Samp_i is computable in time $\widetilde{O}(n)$. Item 3 takes time $t \cdot \widetilde{O}(n) = \widetilde{O}(n)$, since Ext_0 and each Ext_i from Lemma 2.14 are computable in time $\widetilde{O}(n)$. Therefore, Ext is computable in overall time $\widetilde{O}(n)$ in this case.

Otherwise, if $k < C\log^2(n/\varepsilon_0)$, then Item 1 takes time $\widetilde{O}(n)$ after a preprocessing step corresponding to finding a primitive element of $\mathbb{F}_q$ with $q \le \mathrm{poly}(n/\varepsilon_0)$. As discussed above, Item 2 takes time $\widetilde{O}(n)$. Item 3 takes time $\widetilde{O}(\ell_0) = \widetilde{O}(n)$ after a preprocessing step, and so Ext is computable in overall time $\widetilde{O}(n)$ after a preprocessing step. Moreover, if the preprocessing step for Ext_0 consists in finding primitive elements of j fields $\mathbb{F}_q$ with orders $q \le \mathrm{poly}(n/\varepsilon_0)$, then by the above the preprocessing step for Ext consists in finding primitive elements of $j+1$ fields $\mathbb{F}_q$ with orders $q \le \mathrm{poly}(n/\varepsilon_0)$. □

Denote by $\log^{(i)}$ the function that iteratively applies log a total of i times (so $\log^{(1)} n = \log n$, $\log^{(2)} n = \log\log n$, and so on). Denote by $\log^*$ the iterated logarithm. Then, we have the following corollary.

Corollary 5.10. *There exists a constant $C > 0$ such that the following holds. Let n be any positive integer and i any positive integer such that $\log^{(i)} n \ge 6C$. Then, for any $k \le n$ and any $\varepsilon \ge n^3 \cdot 2^{-k/2^{C\cdot i}}$ there exists a strong (k, ε)-seeded extractor $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with seed length $d \le C\log^{(i)} n \cdot \log(n/\varepsilon)$ and output length $m \ge k/2^{C\cdot i}$. Furthermore,*

1. *if* $k \geq 2^{C \cdot i} \cdot \log^2(n/\varepsilon)$*, then* Ext *is computable in time* $\widetilde{O}(n)$*;*
2. *if* $k < 2^{C \cdot i} \cdot \log^2(n/\varepsilon)$*, then* Ext *is computable in time* $\widetilde{O}(n)$ *after a preprocessing step which corresponds to finding primitive elements of* i *fields* $\mathbb{F}_q$ *of orders* $q \leq \mathrm{poly}(n/\varepsilon)$ *powers of* 2*.*

Consequently, if we choose i *to be the largest integer such that* $\log^{(i)} n \geq 6C$ *(which satisfies* $i \leq \log^* n$*) we get a strong* (k, ε)*-seeded extractor* $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *with seed length* $d \leq 6C^2 \log(n/\varepsilon)$ *and output length* $m \geq k/2^{C \log^* n}$ *for any error* $\varepsilon \geq n^3 \cdot 2^{-k/2^{C \log^* n}}$*. If* $k \geq 2^{C \log^* n} \cdot \log^2(n/\varepsilon)$*, then* Ext *is computable in time* $\widetilde{O}(n)$*. Otherwise,* Ext *is computable in time* $\widetilde{O}(n)$ *after a preprocessing step which corresponds to finding primitive elements of* $i \leq \log^* n$ *fields* $\mathbb{F}_q$ *of orders* $q \leq \mathrm{poly}(n/\varepsilon)$*.*

Proof. We iteratively apply Lemma 5.9 i times. Let $c, C > 0$ be the constants guaranteed by Lemma 5.9. For the first application of the lemma, we take $\mathsf{Ext}_0\colon \{0,1\}^n \times \{0,1\}^{d_0} \to \{0,1\}^{m_0}$ to be the strong $(k_0 = 0.7n, \varepsilon_0)$ extractor from Lemma 2.14 with $m_0 = k_0/20$ and $\varepsilon_0 \geq 2^{-ck_0/100}$ to be defined later. The corresponding seed length is $d_0 \leq 2m_0 + 4\log(n/\varepsilon_0) + 4$, which satisfies $d_0 \leq k_0$, and so the initial value of u is $u_0 = d_0/\log(n/\varepsilon_0) \leq k_0$. Denote by Ext_1 the resulting strong seeded extractor. In the second application of Lemma 5.9, we instantiate Ext_0 with Ext_1 instead to obtain a new strong seeded extractor Ext_2, and so on. For each $j \in [i]$, we obtain a family of strong (k, ε_j)-seeded extractors $\mathsf{Ext}_j\colon \{0,1\}^n \times \{0,1\}^{d_j} \to \{0,1\}^{m_j}$ parameterized by k with output length $m_j = m_{j-1}(ck)$, error

$$\varepsilon_j = C \log u_{j-1} \cdot \varepsilon_{j-1}(ck)$$

and seed length

$$d_j = C \log u_{j-1} \cdot \log(n/\varepsilon_{j-1}(ck)) = C \log u_{j-1} \cdot \log\left(\frac{n \cdot C \log u_{j-1}}{\varepsilon_j}\right),$$

where

$$\begin{aligned} u_j &= \frac{d_j}{\log(n/\varepsilon_j)} \\ &= C \log u_{j-1} \cdot \left(1 + \frac{\log C}{\log(n/\varepsilon_j)} + \frac{\log\log u_{j-1}}{\log(n/\varepsilon_j)}\right) \\ &\leq C \log u_{j-1} \cdot \left(1 + \frac{\log C}{\log n} + \frac{\log\log u_{j-1}}{\log n}\right) \\ &\leq 3C \log u_{j-1}. \end{aligned}$$

The last inequality uses the fact that $u_{j-1} \leq u_0 \leq n$ for all j.

Recall that from the corollary statement that i is such that $\log^{(i)} n \geq 6C$. We show by induction that $u_j \leq 3C \log^{(j)} n + 3C \log(6C)$ for all $j = 0, \ldots, i$. This is immediate for the base case $j = 0$, since $u_0 \leq k_0 \leq n$. For the induction step, note that

$$\begin{aligned} u_{j+1} \leq 3C \log u_j &\leq 3C \log(3C \log^{(j)} n + 3C \log(6C)) \\ &\leq 3C \log(2 \cdot 3C \log^{(j)} n) = 3C \log^{(j+1)} n + 3C \log(6C), \end{aligned}$$

as desired. This implies that

$$d_j = u_j \cdot \log(n/\varepsilon_j) \leq 6C \log^{(j)} n \cdot \log(n/\varepsilon_j)$$

and

$$\varepsilon_j = C \log u_{j-1} \cdot \varepsilon_{j-1}(ck) \le (6C)^j \left(\prod_{j'=0}^{j-1} \log^{(j')} n \right) \cdot \varepsilon_0(c^j k)$$

for all $j \in [i]$. We may assume that C is large enough that $\log a \le \sqrt{a}$ for all $a \ge C$, in which case $\prod_{j'=0}^{j-1} \log^{(j')} n \le \prod_{j'=0}^{j-1} n^{2^{-j'}} \le n^2$ since $\log^{(j')} n \ge C$ for all $j' \le i$ by hypothesis. Therefore, we obtain final output length

$$m_i = m_0(c^i k) = k/2^{O(i)},$$

final error ε_i satisfying

$$\varepsilon_0(ck) \le \varepsilon_i \le (6C)^i \cdot n^2 \cdot \varepsilon_0(c^i k) \le n^3 \cdot \varepsilon_0(c^i k),$$

where the last inequality uses that $\log^{(i)} n \ge 6C$, and final seed length

$$d_i \le 6C \log^{(i)} n \cdot \log(n/\varepsilon_i).$$

We now instantiate $\varepsilon_0(c^i k) = \varepsilon/n^3$. Note that $\varepsilon_0(c^i k) \ge 2^{-0.7c^{i+1}k/100}$ as required for the choice of Ext_0 above so long as $\varepsilon \ge n^3 \cdot 2^{-0.7c^{i+1}k}$, which holds by the corollary's hypothesis if C is a large enough constant. With this choice of $\varepsilon_0(c^i k)$ we get final error $\varepsilon_i \le n^3 \cdot \varepsilon_0(c^i k) = \varepsilon$. In fact, we can make ε_i larger so that $\varepsilon_i = \varepsilon$, in which case the final seed length satisfies

$$d_i \le 6C \log^{(i)} n \cdot \log(n/\varepsilon),$$

as desired.

Time complexity. Finally, we discuss the time complexity of Ext. Note that the initial choice for Ext_0 is computable in time $\widetilde{O}(n_0)$. Therefore, if $k \ge 2^{C \cdot i} \log^2(n/\varepsilon)$ for a sufficiently large constant $C > 0$, then the conditions of Item 1 of Lemma 5.9 are satisfied for all i applications of this lemma, and so Ext will be computable in time $\widetilde{O}(n)$. Otherwise, the condition in Item 1 holds and so Ext is computable in time $\widetilde{O}(n)$ after a preprocessing step, since we always have $u \le n$ in each application of the lemma. By Lemma 5.9, the preprocessing amounts to finding primitive elements of i fields $\mathbb{F}_q$ with orders $q \le \mathrm{poly}(n/\varepsilon_0) = \mathrm{poly}(n/\varepsilon)$. □

To obtain our final theorem, we use block source extraction to increase the output length of the extractor from Corollary 5.10, following a strategy of Zuckerman [Zuc97].

Theorem 5.11. *There exist constants $c, C > 0$ such that the following holds. For any integers n and $k \le n$ and any $\varepsilon \ge Cn^3 \cdot 2^{-k/\log k}$ there exists a strong (k, ε)-seeded extractor $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with seed length $d \le C \log(n/\varepsilon)$ and output length $m \ge ck$. Furthermore,*

1. *if $k \ge 2^{C \log^* n} \cdot \log^2(n/\varepsilon)$, then Ext is computable in time $\widetilde{O}(n)$;*
2. *if $k < 2^{C \log^* n} \cdot \log^2(n/\varepsilon)$, then Ext is computable in time $\widetilde{O}(n)$ after a preprocessing step which corresponds to finding $\log^* n$ primitive elements of fields $\mathbb{F}_q$ of orders $q \le \mathrm{poly}(n/\varepsilon)$ powers of 2.*

Proof. Define $\varepsilon' = \varepsilon/6$ and let X be an arbitrary (n, k)-source. The extractor Ext behaves as follows on input X:

1. Apply a strong $(k, k', (\varepsilon')^2)$-condenser $\mathsf{Cond}\colon \{0,1\}^n \times \{0,1\}^{d_{\mathsf{Cond}}} \to \{0,1\}^{n'}$ to X, with output min-entropy rate $k' \geq 0.95n'$ and seed length $d_{\mathsf{Cond}} = C_{\mathsf{Cond}} \log(n/\varepsilon')$. If $k \geq 2^{C \log^* n} \cdot \log^2(n/\varepsilon)$, we instantiate Cond with the lossless KT strong (k, k', ε')-condenser (Theorem 3.10). Otherwise, we instantiate Cond with the lossy RS strong (k, k', ε')-condenser (Theorem 3.11). By the second part of either Theorem 3.10 or Theorem 3.11, we get that with probability at least $1 - \varepsilon'$ over the choice of the seed y we obtain an output X' that is ε'-close to an (n', k')-source with $k' \geq 0.95n'$. As in previous arguments, we work under such a good fixing of y from here onwards and account for the probability ε' of selecting a bad seed in the final extractor error later on.

2. Write $X' = X_1 \circ X_2$ with $|X_1| = |X_2| = n'/2$. Choose the constant $c > 0$ in the theorem statement small enough so that $\log(1/\varepsilon') \leq \log(1/\varepsilon) + 3 \leq ck + 3 \leq 0.05k$, which means that $n'/2 - 0.05k - \log(1/\varepsilon') \geq 0.4n'$. Then, combining Item 1 with Lemma 2.24, (instantiated with $t = 2$, $\Delta = 0.05k$, and $\varepsilon = \varepsilon'$) via the triangle inequality, X' is $3\varepsilon'$-close to an $((n'/2, n'/2), 0.8)$-block source.

3. Apply block source extraction to $X_1 \circ X_2$. More precisely, let $\mathsf{Ext}_1\colon \{0,1\}^{n_1} \times \{0,1\}^{d_1} \to \{0,1\}^{m_1}$ be the strong $(k_1 = 0.8n_1, \varepsilon_1)$-seeded extractor from Corollary 5.10 instantiated with $i = 2$ and $n_1 = n'/2$, which yields $\varepsilon_1 = \varepsilon \geq n_1^3 \cdot 2^{-c_1 k_1}$, $d_1 \leq C_1 \log\log k_1 \cdot \log(n'/\varepsilon)$, and $m_1 \geq c_1 k_1$, for constants $c_1, C_1 > 0$ guaranteed by Corollary 5.10. Furthermore, let $\mathsf{Ext}_2\colon \{0,1\}^{n_2} \times \{0,1\}^{d_2} \to \{0,1\}^{m_2}$ be the strong $(k_2 = 0.8n_2, \varepsilon_2)$-seeded extractor from the "Consequently" part of Corollary 5.10 and $n_2 = n'/2$, which yields $\varepsilon_2 = n_2^3 \cdot 2^{-k_2/2^{C_2 \log^* k_2}}$, $d_2 \leq C_2 \log(n'/\varepsilon)$, and $m_2 \geq k_2/2^{C_2 \log^* k_2}$, for a constant $C_2 > 0$ guaranteed by Corollary 5.10. This choice of parameters ensures that $m_2 \geq d_1$. Indeed, since $k \geq k_1 = k_2 \geq 0.4n'$, to see that $m_2 \geq d_1$ it suffices to check that

$$\frac{0.4k}{2^{C_2 \log^* k}} \geq d_1 = C_1 \log\log k \cdot \log(n'/\varepsilon_1).$$

Since $\varepsilon_1 = \varepsilon' = \varepsilon/5$ and $\log(n'/\varepsilon_1) = O(\log(k/\varepsilon')) = O(\log k + k/\log k) = O(k/\log k)$, it is enough that

$$k \geq C_1' \cdot 2^{C_2 \log^* k} \log\log k \cdot \frac{k}{\log k}$$

for a sufficiently large constant $C_1' > 0$, which holds whenever k is larger than some appropriate absolute constant. Instantiating Lemma 2.23 with Ext_1 and Ext_2 above yields a strong $(k_1 = 0.8n_1, k_2 = 0.8n_2, \varepsilon_1 + \varepsilon_2)$-block-source extractor $\mathsf{BExt}\colon \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \{0,1\}^{d_2} \to \{0,1\}^{m_1}$.

Since X' is $3\varepsilon'$-close to an $(n_1, n_2, 0.8)$-block source, we conclude that

$$Y_{\mathsf{BExt}} \circ \mathsf{BExt}(X', Y_{\mathsf{BExt}}) \approx_{3\varepsilon' + \varepsilon_1 + \varepsilon_2} U_{d_2 + m_1}. \tag{9}$$

We define the output of our final strong extractor $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{m_1}$ to be $\mathsf{BExt}(X', Y_{\mathsf{BExt}})$. Since $\varepsilon_1, \varepsilon_2 \leq \varepsilon'$, Equation (9) implies that

$$Y_{\mathsf{Cond}} \circ Y_{\mathsf{BExt}} \circ \mathsf{Ext}(X, Y_{\mathsf{Cond}} \circ Y_{\mathsf{BExt}}) \approx_{5\varepsilon'} U_{d + m_1}.$$

This means that Ext is a strong $(k, \varepsilon' + 5\varepsilon' = \varepsilon)$-seeded extractor with seed length $d = |Y_{\mathsf{Cond}}| + |Y_{\mathsf{BExt}}| = O(\log(n/\varepsilon))$ and output length $m_1 \geq c_1 k_1 \geq c_1' k$ for an absolute constant $c_1' > 0$, where one of the ε' terms in the error comes from fixing the seed in the condensing step of Item 1.

Time complexity. Finally, we analyze the time complexity of Ext. If $k \geq 2^{C \log^* n} \cdot \log^2(n/\varepsilon)$, then Item 1 runs in time $\widetilde{O}(n)$. In Item 3, Ext_1 and Ext_2 are both computable in time $\widetilde{O}(n)$ under this lower bound on k, and thus so is BExt. We conclude that Ext runs in time $\widetilde{O}(n)$. Otherwise, if $k < 2^{C \log^* n} \cdot \log^2(n/\varepsilon)$, then Item 1 runs in time $\widetilde{O}(n)$ after a preprocessing step, and Ext_1 and Ext_2 in Item 3 run in time $\widetilde{O}(n)$ after a preprocessing step. Therefore, overall, Ext runs in time $\widetilde{O}(n)$ after a preprocessing step. □

References

[ACG+22] Omar Alrabiah, Eshan Chattopadhyay, Jesse Goodman, Xin Li, and João Ribeiro. Low-degree polynomials extract from local sources. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 10:1–10:20, 2022.

[AGHP92] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992.

[AGMR24] Omar Alrabiah, Jesse Goodman, Jonathan Mosheiff, and João Ribeiro. Low-degree polynomials are good extractors. *ECCC*, 2024. https://eccc.weizmann.ac.il/report/2024/093/ (manuscript).

[Alo21] Noga Alon. Explicit expanders of every degree and size. *Combinatorica*, pages 1–17, 2021.

[BBCM95] Charles H. Bennett, Gilles Brassard, Claude Crépeau, and Ueli M. Maurer. Generalized privacy amplification. *IEEE Transactions on Information Theory*, 41(6):1915–1923, 1995.

[BG13] Andrej Bogdanov and Siyao Guo. Sparse extractor families for all the entropy. In *Innovations in Theoretical Computer Science (ITCS)*, pages 553–560. ACM, 2013.

[BM74] Allan Borodin and Robert Moenck. Fast modular transforms. *Journal of Computer and System Sciences*, 8(3):366–386, 1974.

[Bog12] Andrej Bogdanov. Topics in (and out) the theory of computing: Lecture notes. https://andrejb.net/csc5060/notes/12L12.pdf, 2012. [Online; accessed October 2024].

[BRST02] Ziv Bar-Yossef, Omer Reingold, Ronen Shaltiel, and Luca Trevisan. Streaming computation of combinatorial objects. In *Annual Conference on Computational Complexity (CCC)*, pages 165–174. IEEE, 2002.

[CG88] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, 1988.

[CL18] Kuan Cheng and Xin Li. Randomness extraction in AC0 and with small locality. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 37:1–37:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[CRSW13] L. Elisa Celis, Omer Reingold, Gil Segev, and Udi Wieder. Balls and bins: Smaller hash families and faster evaluation. *SIAM Journal on Computing*, 42(3):1030–1050, 2013.

[CT65] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[CW24] Kuan Cheng and Ruiyang Wu. Randomness extractors in AC^0 and NC^1: Optimal up to constant factors. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 69:1–69:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[DD06] Jacques Dubrois and Jean-Guillaume Dumas. Efficient polynomial time algorithms computing industrial-strength primitive roots. *Information Processing Letters*, 97(2):41–45, 2006.

[DKSS13] Zeev Dvir, Swastik Kopparty, Shubhangi Saraf, and Madhu Sudan. Extensions to the method of multiplicities, with applications to Kakeya sets and mergers. *SIAM Journal on Computing*, 42(6):2305–2328, 2013.

[DMOZ22] Dean Doron, Dana Moshkovitz, Justin Oh, and David Zuckerman. Nearly optimal pseudorandomness from hardness. *J. ACM*, 69(6), November 2022.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008.

[DPVR12] Anindya De, Christopher Portmann, Thomas Vidick, and Renato Renner. Trevisan's extractor in the presence of quantum side information. *SIAM Journal on Computing*, 41(4):915–940, 2012.

[DT23] Dean Doron and Roei Tell. Derandomization with minimal memory footprint. In *Computational Complexity Conference (CCC)*, pages 11:1–11:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[Für09] Martin Fürer. Faster integer multiplication. *SIAM Journal on Computing*, 39(3):979–1005, 2009.

[FWE+23] Cameron Foreman, Sherilyn Wright, Alec Edgington, Mario Berta, and Florian J. Curchod. Practical randomness amplification and privatisation with implementations on quantum computers. *Quantum*, 7:969, March 2023.

[FYEC24] Cameron Foreman, Richie Yeung, Alec Edgington, and Florian J. Curchod. Cryptomite: A versatile and user-friendly library of randomness extractors. *arXiv e-prints*, February 2024. https://arxiv.org/abs/2402.09481.

[GGH+24] Alexander Golovnev, Zeyu Guo, Pooya Hatami, Satyajeet Nagargoje, and Chao Yan. Hilbert functions and low-degree randomness extractors. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 41:1–41:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[Gil98] David Gillman. A Chernoff bound for random walks on expander graphs. *SIAM Journal on Computing*, 27(4):1203–1220, 1998.

[GUV09] Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from Parvaresh–Vardy codes. *J. ACM*, 56(4), jul 2009.

[GVW15] Oded Goldreich, Emanuele Viola, and Avi Wigderson. On randomness extraction in AC0. In *Conference on Computational Complexity (CCC)*, page 601–668. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[Hea08] Alexander D. Healy. Randomness-efficient sampling within NC^1. *Computational Complexity*, 17:3–37, 2008.

[HH15] Jan Hązła and Thomas Holenstein. Upper tail estimates with combinatorial proofs. In *Symposium on Theoretical Aspects of Computer Science (STACS)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[HIV22] Xuangui Huang, Peter Ivanov, and Emanuele Viola. Affine extractors and AC0-parity. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 9:1–9:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[HT16] Masahito Hayashi and Toyohiro Tsurumaru. More efficient privacy amplification with less random seeds via dual universal hash function. *IEEE Transactions on Information Theory*, 62(4):2213–2232, 2016.

[HV06] Alexander Healy and Emanuele Viola. Constant-depth circuits for arithmetic in finite fields of characteristic two. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 672–683. Springer, 2006.

[HvdH19] David Harvey and Joris van der Hoeven. Faster polynomial multiplication over finite fields using cyclotomic coefficient rings. *Journal of Complexity*, 54:101404, 2019.

[HvdH21] David Harvey and Joris van der Hoeven. Integer multiplication in time $O(n\log n)$. *Annals of Mathematics*, 193(2):563 – 617, 2021.

[HvdH22] David Harvey and Joris van der Hoeven. Polynomial multiplication over finite fields in time $O(n\log n)$. *J. ACM*, 69(2):1–40, 2022.

[Jus72] Jørn Justesen. Class of constructive asymptotically good algebraic codes. *IEEE Transactions on Information Theory*, 18(5):652–656, 1972.

[KT22] Itay Kalev and Amnon Ta-Shma. Unbalanced expanders from multiplicity codes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 12:1–12:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[Li15] Xin Li. Non-malleable condensers for arbitrary min-entropy, and almost optimal protocols for privacy amplification. In *Theory of Cryptography Conference (TCC)*, pages 502–531. Springer, 2015.

[LRVW03] Chi-Jen Lu, Omer Reingold, Salil Vadhan, and Avi Wigderson. Extractors: Optimal up to constant factors. In *Symposium on Theory of Computing (STOC)*, pages 602–611. ACM, 2003.

[Lu02] Chi-Jen Lu. Hyper-encryption against space-bounded adversaries from on-line strong extractors. In *Advances in Cryptology — CRYPTO*, pages 257–271. Springer, 2002.

[MPS12] Wolfgang Mauerer, Christopher Portmann, and Volkher B. Scholz. A modular framework for randomness extraction based on Trevisan's construction. *arXiv e-prints*, December 2012. https://arxiv.org/abs/1212.0520.

[MRRR14] Raghu Meka, Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Fast pseudorandomness for independence and load balancing. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 859–870. Springer, 2014.

[MW97] Ueli Maurer and Stefan Wolf. Privacy amplification secure against active adversaries. In *Advances in Cryptology — CRYPTO*, pages 307–321. Springer, 1997.

[NN90] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. In *Symposium on Theory of Computing (STOC)*, pages 213–223. ACM, 1990.

[NT99] Noam Nisan and Amnon Ta-Shma. Extracting randomness: A survey and new constructions. *Journal of Computer and System Sciences*, 58(1):148–173, 1999.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.

[Rao07] Anup Rao. An exposition of Bourgain's 2-source extractor. *ECCC*, 2007. https://eccc.weizmann.ac.il//report/2007/034/ (manuscript).

[RRV02] Ran Raz, Omer Reingold, and Salil Vadhan. Extracting all the randomness and reducing the error in Trevisan's extractors. *Journal of Computer and System Sciences*, 65(1):97–128, 2002.

[RSW06] Omer Reingold, Ronen Shaltiel, and Avi Wigderson. Extracting randomness via repeated condensing. *SIAM Journal on Computing*, 35(5):1185–1209, 2006.

[RT00] Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Discrete Mathematics*, 13(1):2–24, 2000.

[Sch77] Arnold Schönhage. Schnelle multiplikation von polynomen über körpern der charakteristik 2. *Acta Informatica*, 7(4):395–398, 1977.

[Sho90] Victor Shoup. Searching for primitive roots in finite fields. In *Symposium on Theory of Computing (STOC)*, pages 546–554. ACM, 1990.

[Shp92] Igor E. Shparlinski. On primitive elements in finite fields and on elliptic curves. *Mathematics of the USSR-Sbornik*, 71(1):41, feb 1992.

[Spi96] Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1731, 1996.

[SU05] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *J. ACM*, 52(2):172–216, 2005.

[SZ99] Aravind Srinivasan and David Zuckerman. Computing with very weak random sources. *SIAM Journal on Computing*, 28(4):1433–1459, 1999.

[Ta-17] Amnon Ta-Shma. Explicit, almost optimal, epsilon-balanced codes. In *Symposium on Theory of Computing (STOC)*, page 238–251. ACM, 2017.

[Tre01] Luca Trevisan. Extractors and pseudorandom generators. *J. ACM*, 48(4):860–879, jul 2001.

[TSSR11] Marco Tomamichel, Christian Schaffner, Adam Smith, and Renato Renner. Leftover hashing against quantum side information. *IEEE Transactions on Information Theory*, 57(8):5524–5535, 2011.

[TU12] Amnon Ta-Shma and Christopher Umans. Better condensers and new extractors from Parvaresh-Vardy codes. In *Conference on Computational Complexity (CCC)*, pages 309–315. IEEE, 2012.

[TZS06] Amnon Ta-Shma, David Zuckerman, and Shmuel Safra. Extractors from Reed–Muller codes. *Journal of Computer and System Sciences*, 72(5):786–812, 2006.

[Vad04] Salil Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *Journal of Cryptology*, 17:43–77, 2004.

[Vad12] Salil Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1–3):1–336, 2012.

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2013.

[WZ99] Avi Wigderson and David Zuckerman. Expanders that beat the eigenvalue bound: Explicit construction and applications. *Combinatorica*, 19(1):125–138, 1999.

[XZ24] Zhiyang Xun and David Zuckerman. Near-optimal averaging samplers. *ECCC*, 2024. https://eccc.weizmann.ac.il/report/2024/097/ (manuscript).

[Zuc97] David Zuckerman. Randomness-optimal oblivious sampling. *Random Structures & Algorithms*, 11(4):345–367, 1997.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il