Randomness Extraction in AC^0 and with Small Locality

Kuan Cheng[1]
Department of Computer Science, Johns Hopkins University.
kcheng17@jhu.edu.

Xin Li[2]
Department of Computer Science, Johns Hopkins University.
lixints@cs.jhu.edu.

Abstract

Randomness extractors, which extract high quality (almost-uniform) random bits from biased random sources, are important objects both in theory and in practice. While there have been significant progress in obtaining near optimal constructions of randomness extractors in various settings, the computational complexity of randomness extractors is still much less studied. In particular, it is not clear whether randomness extractors with good parameters can be computed in several interesting complexity classes that are much weaker than P.

In this paper we study randomness extractors in the following two models of computation: (1) constant-depth circuits (AC^0), and (2) the local computation model. Previous work in these models, such as [38], [15] and [6], only achieve constructions with weak parameters. In this work we give explicit constructions of randomness extractors with much better parameters. Our results on AC^0 extractors refute a conjecture in [15] and answer several open problems there. We also provide a lower bound on the error of extractors in AC^0, which together with the entropy lower bound in [38, 15] almost completely characterizes extractors in this class. Our results on local extractors also significantly improve the seed length in [6]. As an application, we use our AC^0 extractors to study pseudorandom generators in AC^0, and show that we can construct both cryptographic pseudorandom generators (under reasonable computational assumptions) and unconditional pseudorandom generators for space bounded computation with very good parameters.

Our constructions combine several previous techniques in randomness extractors, as well as introduce new techniques to reduce or preserve the complexity of extractors, which may be of independent interest. These include (1) a general way to reduce the error of strong seeded extractors while preserving the AC^0 property and small locality, and (2) a seeded randomness condenser with small locality.

2012 ACM Subject Classification Theory of computation → Expander graphs and randomness extractors, Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Randomness Extraction, AC0, Locality, Pseudorandom Generator

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2018.37

Related Version A full version of this paper appears in [10], https://arxiv.org/abs/1602.01530.

[1] Supported in part by NSF award CCF-1617713.
[2] Supported in part by NSF award CCF-1617713.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018).
Editors: Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer; Article No. 37; pp. 37:1–37:20

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Randomness extractors are functions that transform biased random sources into almost uniform random bits. Throughout this paper, we model biased random sources by the standard model of general weak random sources, which are probability distributions over n-bit strings with a certain amount of min-entropy k.[3] Such sources are referred to as (n, k)-sources. In this case, it is well known that no deterministic extractors can exist for one single weak random source even if $k = n - 1$; therefore seeded randomness extractors were introduced in [32], which allow the extractors to have a short uniform random seed (say length $O(\log n)$). In typical situations, we require the extractor to be *strong* in the sense that the output is close to uniform even given the seed. Formally, we have the following definition.

▶ **Definition 1** ([32]). A function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a seeded (k, ϵ) extractor if for any (n, k) source X, we have

$$|\mathsf{Ext}(X, U_d) - U_m| \leq \epsilon.$$

Ext is strong if in addition $|(\mathsf{Ext}(X, U_d), U_d) - (U_m, U_d)| \leq \epsilon$, where U_m and U_d are independent uniform strings on m and d bits respectively, and $|\cdot|$ stands for the statistical distance.

Since their introduction, seeded randomness extractors have become fundamental objects in pseudorandomness, and have found numerous applications in derandomization, complexity theory, cryptography and many other areas in theoretical computer science. In addition, through a long line of research, we now have explicit constructions of seeded randomness extractors with almost optimal parameters (e.g., [17]). However, the complexity of randomness extractors is still much less studied and understood. For example, while in general explicit constructions of randomness extractors can be computed in polynomial time of the input size, some of the known constructions are actually more explicit than that. These include for example extractors based on universal hashing [8], and Trevisan's extractor [36], which can be computed by highly uniform constant-depth circuits of polynomial size with parity gates. Thus a main question one can ask is: can we do better and construct good randomness extractors with very low complexity?

This question is interesting not just by its own right, but also because such extractors, as building blocks, can be used to potentially reduce the complexity of other important objects. In this paper we study this question and consider the parallel and local complexity of randomness extractors.

The parallel-AC^0 model. The hierarchy of NC and AC circuits are standard models for parallel computation. It is easy to see that the class of NC^0 or even ℓ-local functions for small ℓ , which correspond to functions where each output bit depends on at most ℓ input bits (including both the weak source and the seed), cannot compute strong extractors (since one can just fix ℓ bits of the source). Thus, a natural relaxation is to consider the class AC^0, which refers to the family of polynomial-size and constant-depth circuits with unbounded fan-in gates. Note that although we have strong lower bounds here for explicit functions, it is still not clear whether some important objects, such as randomness extractors and pseudorandom generators, can be computed in AC^0 with good parameters. Thus the study of this question also helps us better understand the power of this class.

[3] A probability distribution is said to have min-entropy k if the probability of getting any element in the support is at most 2^{-k}.

Viola [38] was the first to consider this question, and his result was generalized by Goldreich et al. [15] to show that for strong seeded extractors, even extracting a single bit is impossible if $k < n/\mathsf{poly}(\log n)$. When $k \geq n/\mathsf{poly}(\log n)$, Goldreich et al. showed how to extract $\Omega(\log n)$ bits using $O(\log n)$ bits of seed, or more generally how to extract $m < k/2$ bits using $O(m)$ bits of seed. Note that the seed length is longer than the output length.[4] When the extractor does not need to be strong, they showed that extracting $r + \Omega(r)$ bits using r bits of seed is impossible if $k < n/\mathsf{poly}(\log n)$; while if $k \geq n/\mathsf{poly}(\log n)$ one can extract $(1 + c)r$ bits for some constant $c > 0$, using r bits of seed. All the positive results here have error $1/\mathsf{poly}(n)$.

Therefore, a natural and main open problem left in [15] is whether one can construct randomness extractors in AC^0 with shorter seed and longer output. Specifically, [15] asks if one can extract more than $\mathsf{poly}(\log n)r$ bits in AC^0 using a seed length $r = \Omega(\log n)$, when $k \geq n/\mathsf{poly}(\log n)$. In [15] the authors conjectured that the answer is negative. Another open question is to see if one can achieve better error, e.g., negligible error instead of $1/\mathsf{poly}(n)$.

Goldreich et al. [15] also studied deterministic extractors for bit-fixing sources, and most of their effort went into extractors for oblivious bit-fixing sources (although they also briefly studied non-oblivious bit-fixing sources). An (n, k)-oblivious bit-fixing source is a string of n bits such that some unknown k bits are uniform, while the other $n - k$ bits are fixed. Extractors for such sources are closely related to exposure-resilient cryptography [7, 24]. In this case, a standard application of Håstad's switching lemma [18] implies that it is impossible to construct extractors in AC^0 for bit-fixing sources with min-entropy $k < n/\mathsf{poly}(\log n)$. The main result in [15] is a theorem which shows the *existence* of deterministic extractors in AC^0 for min-entropy $k \geq n/\mathsf{poly}(\log n)$ that output $k/\mathsf{poly}(\log n)$ bits with error $2^{-\mathsf{poly}(\log n)}$. We emphasize that this is an existential result, and [15] did not give any explicit constructions of such extractors.

The local model. Another relaxation, introduced by Bogdanov and Guo [6], is the notion of sparse extractor families. These are families of functions for which each function in the family has a small number of overall input-output dependencies (referred to as the sparsity, meaning that the input-output dependency graph is sparse), while taking a random function from the family serves as a randomness extractor. Such extractors can be used generally in situations where hashing is used and preserving small input-output dependencies is needed. As an example, the authors in [6] used such extractors to obtain a transformation of non-uniform one-way functions into non-uniform pseudorandom generators that preserves output locality.

In this paper, we consider the condition of the family being ℓ-local, which is a worst case notion rather than the average case notion of sparsity. Furthermore, we will focus on the case of strong extractor families. Note that a strong extractor family is equivalent to a strong seeded extractor, since the randomness used to choose a function from the family can be included in the seed. Thus, we study strong seeded extractors with small locality, i.e., for any fixing of the seed, each output bit depends on at most ℓ input bits.

Note that an extractor family with m output bits and locality ℓ is automatically an ℓm-sparse extractor family. Conversely, if an extractor is s-sparse, then half of its output bits depend on at most $2s/m$ input bits, so by removing half of the output bits one could obtain locality $2s/m$; a technical point here is that one may need to drop different output bits depending on the seed, but this does not affect the error of the extractor.

[4] They also showed how to extract $\mathsf{poly}(\log n)$ bits using an $O(\log n)$ bit seed, but the error of the extractor becomes $1/\mathsf{poly}(\log n)$.

The authors of [6] gave a construction of a strong extractor family for all entropy k with output length $m \leq k$, error ϵ, and sparsity $O(n \log(m/\epsilon) \log(n/m))$, which corresponds to locality $O(\frac{n}{m} \log(\frac{m}{\epsilon}) \log(\frac{n}{m})) = \Omega(n/k \log(n/\epsilon))$ whenever $k \leq n/2$. They also showed that such sparsity is necessary whenever $n^{0.99} \leq m \leq n/6$ and ϵ is a constant. However, the main drawback of the construction in [6] is that the family size is quite large. Indeed the family size is 2^{nm}, which corresponds to a seed length of at least nm.[5] Therefore, a main open problem in [6] is to reduce the size of the family (or, equivalently, the seed length).

De and Trevisan [11] obtained a strong extractor for (n, k) sources such that for any fixing of the seed, each bit of the extractor's output only depends on $\mathsf{poly}(\log n)$ bits of the source. However, their construction only works for $k = \delta n$ where δ is any constant. Their extractor has seed length $d = O(\log n)$ and outputs $k^{\Omega(1)}$ bits, but the error is only $n^{-\alpha}$ for a small constant $0 < \alpha < 1$.

It is also worthwhile to compare our definition of a strong extractor family with small locality to the definition of t-local extractors given by Vadhan [37]. For a t-local extractor, one requires that for any fixing of the seed r, the output of the function $\mathsf{Ext}(x, r)$ *as a whole* depends on only t bits of x. In contrast, our definition requires that *each output bit* of the function $\mathsf{Ext}(x, r)$ depends on at most ℓ bits of x. It can be seen that a strong extractor with sparsity t is automatically a t-local extractor, but the converse may not be true: a t-local extractor may have locality t and sparsity up to mt. By a lower bound in [37], the parameter t in local-extractors is at least $\Omega(nm/k)$, which is larger than m and matches the sparsity in [6] and our results up to polylogarithmic factors. In this sense, our definition of extractor with small locality is stronger than t-local extractors. Furthermore, the construction of t-local extractors in [37], which uses the sample-then-extract approach, only works for large min-entropy (at least $k > \sqrt{n}$); while our goal here is to construct strong extractor families even for very small min-entropy, with locality $\ell \ll m$.

Our results

As in [38, 15], in this paper we obtain both negative results and positive results about randomness extraction in AC^0. While the negative results in [38, 15] provide lower bounds on the entropy required for AC^0 extractors, our negative results provide lower bounds on the error such extractors can achieve. We show that such extractors (both seeded extractors and deterministic extractors for bit-fixing sources) cannot achieve error better than $2^{-\mathsf{poly}(\log n)}$, even if the entropy of the sources is quite large. Specifically, we have

▶ **Theorem 2.** *(General weak source) If* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a strong* $(k = n - 1, \epsilon)$*-extractor that can be computed by* AC^0 *circuits of depth* $\mathtt{dth}$ *and size* s*, then* $\epsilon = 2^{-(O(\log s))^{\mathtt{dth}-1} \log(n+d)}$.

(Bit-fixing source) There is a constant $c > 1$ *such that if* $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^m$ *is a* (k, ϵ)*-extractor for oblivious bit-fixing sources with* $k = n - (c \log s)^{\mathtt{dth}-1}$*, that can be computed by* AC^0 *circuits of depth* $\mathtt{dth}$ *and size* s*, then* $\epsilon = 2^{-(O(\log s))^{\mathtt{dth}-1} \log n}$. [6]

Thus, our results combined with the lower bounds on the entropy requirement in [38, 15] almost completely characterize the power of randomness extractors in AC^0.

We now turn to our positive results. As our first contribution, we show that the authors' conjecture about seeded AC^0 extractors in [15] is false. We give explicit constructions of *strong* seeded extractors in AC^0 with much better parameters. This in particular answers open problems 8.1 and 8.2 in [15]. To start with, we have the following theorem.

[5] In fact, the seed length is even larger since the seed is used to sample from a non-uniform distribution.
[6] This holds even if we allow Ext to have a uniform random seed, see in the full version [10].

▶ **Theorem 3.** *For any constant $c \in \mathbb{N}$, any $k = \Omega(n/\log^c n)$ and any $\epsilon = 1/\mathsf{poly}(n)$, there exists an explicit construction of a strong (k,ϵ)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ that can be computed by an AC^0 circuit of depth $c+10$, where $d = O(\log n)$, $m = k^{\Omega(1)}$ and the extractor family has locality $O(\log^{c+5} n)$.*

Note that the depth of the circuit is almost optimal, within an additive $O(1)$ factor of the lower bound given in [15]. In addition, our construction is also a family with locality only $\mathsf{poly}(\log n)$. Note that the seed length $d = O(\log n)$ is (asymptotically) optimal, while the locality beats the one obtained in [6] (which is $O(n/m \log(m/\epsilon)\log(n/m)) = n^{\Omega(1)}$) and is within a $\log^4 n$ factor to $O(n/k \log(n/\epsilon))$.

Our result also improves that of De and Trevisan [11], even in the high min-entropy case, as our error can be any $1/\mathsf{poly}(n)$ instead of just $n^{-\alpha}$ for some constant $0 < \alpha < 1$. Moreover, our seed length remains $O(\log n)$ even for $k = n/\mathsf{poly}(\log n)$, while in this case the extractor in [11] has seed length $\mathsf{poly}(\log n)$.

Next, we can boost our construction to reduce the error and extract almost all the entropy. We have

▶ **Theorem 4.** *For any constant $\gamma \in (0,1)$, $a, c \in \mathbb{N}$, any $k = \delta n = \Omega(n/\log^c n)$, $\epsilon = 1/2^{O(\log^a n)}$, there exists an explicit strong (k,ϵ)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ in AC^0 with depth $O(a+c+1)$ where $d = O((\log n + \frac{\log(n/\epsilon)\log(1/\epsilon)}{\log n})/\delta)$, $m = (1-\gamma)k$.*

As our second contribution, we give *explicit* deterministic extractors in AC^0 for oblivious bit-fixing sources with entropy $k \geq n/\mathsf{poly}(\log n)$, which output $(1-\gamma)k$ bits with error $2^{-\mathsf{poly}(\log n)}$. This is in contrast to the non-explicit existential result in [15]. Further, the output length and error of our extractor are almost optimal, while the output length in [15] is only $k/\mathsf{poly}(\log n)$. Specifically, we have

▶ **Theorem 5.** *For any constant $a, c \in \mathbb{N}$ and any constant $\gamma \in (0,1]$, there exists an explicit deterministic $(k = \Omega(n/\log^a n), \epsilon = 2^{-\log^c n})$-extractor $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^{(1-\gamma)k}$ that can be computed by AC^0 circuits of depth $O(a+c+1)$, for any (n,k)-bit-fixing source.*

For sparse extractor families, we can reduce the error of Theorem 3 while keeping the locality small.

▶ **Theorem 6.** *There exists a constant $\alpha \in (0,1)$ such that for any $k \geq \frac{n}{\mathsf{poly}(\log n)}$ and $\epsilon \geq 2^{-k^\alpha}$, there exists an explicit construction of a strong (k,ϵ)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, with $d = O(\log n + \frac{\log(n/\epsilon)\log(1/\epsilon)}{\log n})$, $m = k^{\Omega(1)}$ and locality $\log^2(1/\epsilon)\mathsf{poly}(\log n)$.*

We also give strong extractor families with small locality for min-entropy k as small as $\log^2 n$. Our approach is to first condense it into another weak source with constant entropy rate. For this purpose we introduce the following definition of a (strong) randomness condenser with small locality.

▶ **Definition 7.** A function $\mathsf{Cond} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{n_1}$ is a strong (n,k,n_1,k_1,ϵ)-condenser if for every (n,k)-source X and independent uniform seed $R \in \{0,1\}^d$, $R \circ \mathsf{Cond}(X,R)$ is ϵ-close to $R \circ D$, where D is a distribution on $\{0,1\}^{n_1}$ such that for any $r \in \{0,1\}^d$, we have that $D|_{R=r}$ is an (n_1,k_1)-source. We say the condenser family has locality ℓ if for every fixing of $R = r$, the function $\mathsf{Cond}(.,r)$ can be computed by an ℓ-local function.

We now have the following theorem.

▶ **Theorem 8.** *For any $k \geq \log^2 n$, there exists a strong $(n, k, t = 10k, 0.08k, \epsilon)$-condenser $\mathsf{Cond} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^t$ with $d = O(k)$, $\epsilon = 2^{-\Omega(k)}$ and locality $O(\frac{n}{k} \log n)$.*

Combining the condenser with our previous extractors, we get strong extractor families with small locality for any min-entropy $k \geq \log^2 n$. Specifically, we have

▶ **Theorem 9.** *There exits a constant $\alpha \in (0,1)$ such that for any $k \geq \log^2 n$, any constant $\gamma \in (0,1)$ and any $\epsilon \geq 2^{-k^\alpha}$, there exists a strong (k, ϵ)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, where $d = O(k), m = (1-\gamma)k$ and the extractor family has locality $\frac{n}{k} \log^2(1/\epsilon)(\log n)\mathsf{poly}(\log k)$.*

In the above two extractors, our seed length is still much better than that of [6]. However, our locality becomes slightly worse.

Related work, independent work and further results

The work of Dziembowski and Maurer [12] also gave extractors with small locality. However, the model of the weak source studied there is uniform random bits subject to a bounded leakage, which is more restrictive than the model of general weak random sources we consider here. In particular, the analysis of [12], which uses a guessing game, may not work for general weak random sources. A recent independent work by Papakonstantinou et. al [34] used similar techniques as [12] and gave constructions of seeded extractors in the multi-stream model [16]. Their main motivation and result is an extractor that can extract $\Omega(k)$ bits from any (n, k) source with $k = \Omega(n)$, using $O(\log n \log(n/\epsilon))$ bits of seed together with two streams, $O(\log\log(n/\epsilon))$ passes and $O(\log(n/\epsilon))$ space. However, it turns out that their construction can also be realized in AC^0 and also has the property of small locality. Specifically, for an (n, k) source with $k = \delta n = n/\mathsf{poly}\log(n)$ and error $\epsilon = 2^{-\mathsf{poly}\log(n)}$, their construction gives an AC^0 extractor with seed length $O(\frac{1}{\delta^{O(1)}} \log n \log(n/\epsilon))$ and locality $O(\frac{1}{\delta^{O(1)}} \log(n/\epsilon))$. Their construction, which is based on randomly re-bucketing the bits of the source into blocks and arguing this results in a block source, can be viewed as orthogonal to our construction, which is based on hardness amplification. Compared to our results (Theorem 4 and Theorem 6), they have a better dependence on ϵ but we have a better dependence on δ and n. For very small entropy (e.g., $k = \log^2 n$), we can first use our condenser and then apply their construction, which will give an extractor with seed length $O(k)$, output length $\Omega(k)$ and locality $O(\frac{n}{k} \log n \log(k/\epsilon))$.

Applications to pseudorandom generators in AC^0

Like extractors, pseudorandom generators are also fundamental objects in the study of pseudorandomness, and constructing "more explicit" pseudorandom generators is another interesting question that has gained a lot of attention. A pseudorandom generator (or PRG for short) is an efficient deterministic function that maps a short random seed into a long output that looks uniform to a certain class of distinguishers.

▶ **Definition 10.** A function $G : \{0,1\}^n \to \{0,1\}^m$ is a pseudorandom generator for a class $\mathcal{C}$ of Boolean functions with error ϵ, if for every function $A \in \mathcal{C}$, we have that

$$|\Pr[A(U_m) = 1] - \Pr[A(G(U_n)) = 1]| \leq \epsilon.$$

Here we mainly consider two kinds of pseudorandom generators, namely cryptographic PRGs, which are necessarily based on computational assumptions; and unconditional PRGs, most notably PRGs for space bounded computation.

Standard cryptographic PRGs (i.e., PRGs that fool polynomial time computation or polynomial size circuits with negligible error) are usually based on one-way functions (e.g., [19]), and can be computed in polynomial time. However, more explicit PRGs have also been considered in the literature, for the purpose of constructing more efficient cryptographic protocols. Impagliazzo and Naor [23] showed how to construct such a PRG in AC^0, which stretches n bits to $n + \log n$ bits. Their construction is based on the assumed intractability of the subset sum problem. On the other hand, Viola [39] showed that there is no black-box PRG construction with linear stretch in AC^0 from one-way functions. Thus, to get such stretch one must use non black-box constructions.

In [3, 4], Applebaum et al. showed that the existence of cryptographic PRGs in NC^0 with sub-linear stretch follows from a variety of standard assumptions, and they constructed a cryptographic PRG in NC^0 with linear stretch based on a specific intractability assumption related to the hardness of decoding sparsely generated linear codes. In [2], Applebaum further constructed PRG *collections* (i.e., a family of PRG functions) with linear stretch and polynomial stretch based on the assumption of one-wayness of a variant of the random local functions proposed by Goldreich [14].

In the case of unconditional PRGs, for $d \geq 5$ Mossel et al. [29] constructed d-local PRGs with output length $n^{\Omega(d/2)}$ that fool all linear tests with error $2^{-n^{\frac{1}{2\sqrt{d}}}}$, which were used by Applebaum et al. [3] to give a 3-local PRG with linear stretch that fools all linear tests. In the same paper, Applebaum et al. also gave a 3-local PRG with sub linear stretch that fools sublinear-space computation. Thus, it remains to see if we can construct better PRGs (cryptographic or unconditional) in NC^0 or AC^0 with better parameters.

Our PRGs. We show that under reasonable computational assumptions, we can construct very good cryptographic PRGs in AC^0 (e.g. with polynomial stretch and negligible error). In addition, we show that we can construct very good unconditional PRGs for space bounded computation in AC^0 (e.g., with polynomial stretch).

We first give explicit cryptographic PRGs in AC^0 based on the one-wayness of random local functions, the same assumption as used in [2]. To state the assumption we first need the following definitions.

▶ **Definition 11** (Hypergraphs [2])**.** An (n, m, d) hypergraph is a graph over n vertices and m hyperedges each of cardinality d. For each hyperedge $S = (i_0, i_1, \ldots, i_{d-1})$, the indices $i_0, i_1, \ldots, i_{d-1}$ are ordered. The hyperedges of G are also ordered. Let G be denoted as $([n], S_0, S_1, \ldots, S_{m-1})$ where for $i = 0, 1, \ldots, m-1$, S_i is a hyperedge.

▶ **Definition 12** (Goldreich's Random Local Function [14])**.** Given a predicate $Q : \{0,1\}^d \to \{0,1\}$ and an (n, m, d) hypergraph $G = ([n], S_0, \ldots, S_{m-1})$, the function $f_{G,Q} : \{0,1\}^n \to \{0,1\}^m$ is defined as follows: for input x, the ith output bit of $f_{G,Q}(x)$ is $f_{G,Q}(x)_i = Q(x_{S_i})$.

For $m = m(n)$, the function collection $F_{Q,n,m} : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^m$ is defined via the mapping $(G, x) \to f_{G,Q}(x)$, where G is sampled randomly by the s bits and x is sampled randomly by the n bits.

For every $k \in \{0,1\}^s$, we also denote $F(k, \cdot)$ as $F_k(\cdot)$.

▶ **Definition 13** (One-wayness of a Collection of Functions)**.** For $\epsilon = \epsilon(n) \in (0, 1)$, a collection of functions $F : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^m$ is an ϵ-one-way function if for every efficient adversary A which outputs a list of $\mathsf{poly}(n)$ candidates and for sufficiently large n's, we have that

$$\Pr_{k,x,y=F_k(x)}[\exists z \in A(k, y), z' \in F_k^{-1}(y), z = z'] < \epsilon,$$

where k and x are independent and uniform.

We now have the following theorem.

▶ **Theorem 14.** *For any d-ary predicate Q, if the random local function $F_{Q,n,m}$ is δ-one-way for some constant $\delta \in (0,1)$, then we have the following results.*

1. *If there exists a constant $\alpha > 0$ such that $m \geq (1+\alpha)n$, then for any constant $c > 1$, there exists an explicit cryptographic PRG $G : \{0,1\}^r \to \{0,1\}^t$ in AC^0, where $t \geq cr$ and the error is negligible*[7]*.*
2. *If there exists a constant $\alpha > 0$ such that $m \geq n^{1+\alpha}$, then for any constant $c > 1$ there exists an explicit cryptographic PRG $G : \{0,1\}^r \to \{0,1\}^t$ in AC^0, where $t \geq r^c$ and the error is negligible.*

As noted in [2], there are several evidence supporting this assumption. In particular, current evidence is consistent with the existence of a δ-one-way random local function $F_{Q,n,m}$ with $m \geq n^{1+\alpha}$ for some constant $\alpha > 0$.

Compared to the constructions in [2], our construction is in AC^0 instead of NC^0. However, our construction has the following advantages.

- We construct a standard PRG instead of a PRG collection, where the PRG collection is a family of functions and one needs to randomly choose one function before any application.
- The construction of a PRG with polynomial stretch in [2] can only achieve polynomially small error, and for negligible error one needs to assume that the random local function cannot be inverted by any adversary with slightly super polynomial running time. Our construction, on the other hand, achieves negligible error while only assuming that the random local function cannot be inverted by any adversary that runs in polynomial time.

Next we give an explicit PRG in AC^0 with polynomial stretch, that fools space bounded computation. It is a straight forward application of our AC^0-extractor to the Nisan-Zuckerman PRG [32].

▶ **Theorem 15.** *For every constant $c \in \mathbb{N}$ and every $m = m(s) = \mathsf{poly}(s)$, there is an explicit PRG $g : \{0,1\}^{r=O(s)} \to \{0,1\}^m$ in AC^0, such that for any randomized algorithm A using space s,*

$$|\Pr[A(g(U_r)) = 1] - \Pr[A(U_m) = 1]| = \epsilon \leq 2^{-\Theta(\log^c s)},$$

where U_r is the uniform distribution of length r, U_m is the uniform distribution of length m.

Compared to the Nisan-Zuckerman PRG [32], our PRG is in $\mathsf{AC0}$, which is more explicit. On the other hand, our error is $2^{-\Theta(\log^c s)}$ for any constant $c > 0$ instead of being exponentially small as in [32]. It is a natural open problem to see if we can reduce the error to exponentially small. We note that this cannot be achieved by simply hoping to improve the extractor, since our negative result shows that seeded extractors in AC^0 cannot achieve error better than $2^{-\mathsf{poly}(\log n)}$.

2 Lower bounds for error parameters of AC^0 extractors

Our negative results about the error of AC^0 extractors follow by a simple application of Fourier analysis and the well known spectrum concentration theorem of AC^0 functions [27]. We present it in Appendix A.

[7] The error $\epsilon : \mathbb{N} \to [0,1]$ is negligible if $\epsilon(n) = n^{-\omega(1)}$.

3 Constructions of AC^0 extractors for general weak sources

We now briefly describe our positive results. For details of constructions and proofs, please refer to the full version [10]. We will extensively use the following two facts: the parity and inner product over $\mathsf{poly}(\log n)$ bits can be computed by AC^0 circuits of size $\mathsf{poly}(n)$; in addition, any Boolean function on $O(\log n)$ bits can be computed by a depth-2 AC^0 circuit of size $\mathsf{poly}(n)$.

3.1 The basic construction

All our constructions are based on a basic construction of a strong extractor in AC^0 for any $k \geq \frac{n}{\mathsf{poly}(\log n)}$ with seed length $d = O(\log n)$ and error $\epsilon = n^{-\Omega(1)}$. This construction is a a modification of the Impagliazzo-Wigderson pseudorandom generator [21], interpreted as a randomness extractor in the general framework found by Trevisan [36]. The IW-generator first takes a Boolean function on $\log n$ bits, applies a series of hardness amplifications to get another Boolean function on $O(\log n)$ bits, and then uses the Nisan-Wigderson generator [31] together with the new Boolean function. The hardness amplification consists of three steps: the first step, developed by Babai et al. [5], is to obtain a mild average-case hard function from a worst-case hard function; the second step involves a constant number of substeps, with each substep amplifying the hardness by using Impagliazzo's hard core set theorem [22]; the third step, developed by Impagliazzo and Wigderson [21], uses a derandomized direct-product generator to obtain a function that can only be predicted with exponentially small advantage.

Trevisan [36] showed that given an (n, k)-source X, if one regards the n bits of X as the truth table of the initial Boolean function on $\log n$ bits and applies the IW-generator, then by setting parameters appropriately one obtains an extractor. The reason is that for any $x \in \mathsf{supp}(X)$ that makes the output of the extractor fail a certain statistical test T, one can "reconstruct" x by showing that it can computed by a small size circuit, when viewing x as the truth table of the function with T gates. Thus the number of such bad elements $x \in \mathsf{supp}(X)$ is upper bounded by the total number of such circuits. This extractor works for any min-entropy $k \geq n^{\alpha}$.

However, this extractor itself is not in AC^0 (which is not surprising since it can handle min-entropy $k \geq n^{\alpha}$). Thus, at least one of the steps in the construction of the IW-generator/extractor is not in AC^0. By carefully examining each step one can see that the only step not in AC^0 is actually the first step of hardness amplification (This was also pointed out by [38]). Indeed, all the other steps of hardness amplification are essentially doing the same thing: obtaining a function f' on $O(\log n)$ bits from another function f on $O(\log n)$ bits, where the output of f' is obtained by taking the inner product over two $O(\log n)$ bit strings s and r. In addition, s is obtained directly from part of the input of f', while r is obtained by using the other part of the input of f' to generate $O(\log n)$ inputs to f and concatenating the outputs. All of these steps can be done in AC^0, assuming f is in AC^0 (note that f here depends on X).

We therefore modify the IW-generator by removing the first step of hardness amplification, and start with the second step of hardness amplification with the source X as the truth table of the initial Boolean function. Thus the initial function f can be computed by using the $\log n$ input bits to select a bit from X, which can be done in AC^0. Therefore the final Boolean function f' can be computed in AC^0. The last step of the construction, which applies the NW-generator, is just computing f' on several blocks of size $O(\log n)$, which certainly is in AC^0. This gives our basic extractor in AC^0.

The analysis is again similar to Trevisan's argument [36]. However, since we have removed the first step of hardness amplification, now for any $x \in \mathsf{supp}(X)$ that makes the output of the extractor to fail a certain statistical test T, we cannot obtain a small circuit that *exactly computes* x. On the other hand, we can obtain a small circuit that can *approximate* x well, i.e., can compute x correctly on $1-\gamma$ fraction of inputs for some $\gamma = 1/\mathsf{poly}(\log n)$. We then argue that the total number of strings within relative distance γ to the outputs of the circuit is bounded, and therefore combining the total number of possible circuits we can again get a bound on the number of such bad elements in $\mathsf{supp}(X)$. A careful analysis shows that our extractor works for any min-entropy $k \geq n/\mathsf{poly}(\log n)$. However, to keep the circuit size small we have to set the output length to be small enough, i.e., n^{α} and set the error to be large enough, i.e., $n^{-\beta}$.

In Appendix B, we describe more details about the basic construction.

3.1.1 Error reduction

We now describe how we reduce the error of the extractor. We will borrow some techniques from the work of Raz et al. [35], where the authors showed a general way to reduce the error of strong seeded extractors. However, the techniques in Raz et al. [35] do not preserve the AC^0 property, thus our techniques are significantly different from theirs. Nevertheless, our starting point is a lemma from [35], which roughly says the following: given any strong seeded (k,ϵ)-extractor Ext with seed length d and output length m, then for any $x \in \{0,1\}^n$ there exists a set $G_x \subset \{0,1\}^d$ of density $1-O(\epsilon)$, such that if X is a source with entropy slightly larger than k, then the distribution $\mathsf{Ext}(X, G_X)$ is very close to having min-entropy $m - O(1)$. Here $\mathsf{Ext}(X, G_X)$ is the distribution obtained by first sampling x according to X, then sampling r uniformly in G_x and outputting $\mathsf{Ext}(x, r)$.

Giving this lemma, we can apply our basic AC^0 extractor with error $\epsilon = n^{-\beta}$ for some t times, each time with fresh random seed, and then concatenate the outputs. By the above lemma, the concatenation is roughly $(O(\epsilon))^t$-close to a source such that one of the output has min-entropy $m - O(1)$ (i.e., a somewhere high min-entropy source). By choosing t to be a large enough constant the $(O(\epsilon))^t$ can be smaller than any $1/\mathsf{poly}(n)$. We now describe how to extract from the somewhere high min-entropy source with error smaller than any $1/\mathsf{poly}(n)$, in AC^0. This is where our construction differs significantly from [35], as there one can simply apply a good extractor for constant entropy rate.

Assume that we have an AC^0 extractor Ext' that can extract from $(m, m-\sqrt{m})$-sources with error any $\epsilon' = 1/\mathsf{poly}(n)$ and output length $m^{1/3}$. Then we can extract from the somewhere high min-entropy source as follows. We use Ext' to extract from each row of the source with fresh random seed, and then compute the XOR of the outputs. We claim the output is $(2^{-m^{\Omega(1)}} + \epsilon')$-close to uniform. To see this, assume without loss of generality that the i'th row has min-entropy $m - O(1)$. We can now fix the outputs of all the other rows, which has a total size of $tm^{1/3} \ll \sqrt{m}$ as long as t is small. Thus, even after the fixing, with probability $1 - 2^{-m^{\Omega(1)}}$, we have that the i'th row has min-entropy at least $m - \sqrt{m}$. By applying Ext' we know that the XOR of the outputs is close to uniform.

What remains is the extractor Ext'. To construct it we divide the source with length m sequentially into $m^{1/3}$ blocks of length $m^{2/3}$. Since the source has min-entropy $m - \sqrt{m}$, this forms a block source such that each block roughly has min-entropy at least $m^{2/3} - \sqrt{m}$ conditioned on the fixing of all previous ones. We can now take a strong extractor Ext'' in AC^0 with seed length $O(\log n)$ and use the same seed to extract from all the blocks, and concatenate the outputs. It suffices to have this extractor output one bit for each block. Such AC^0 extractors are easy to construct since each block has high min-entropy rate (i.e., $1 - o(1)$). For example, we can use the extractors given by Goldreich et al. [15].

It is straightforward to check that our construction is in AC^0, as long as the final step of computing the XOR of t outputs can be done in AC^0. For error $1/\mathsf{poly}(n)$, it suffices to take t to be a constant and the whole construction is in AC^0, with seed length $O(\log n)$. We can even take t to be $\mathsf{poly}(\log n)$, which will give us error $2^{-\mathsf{poly}(\log n)}$.

3.1.2 Increasing output length

The error reduction step reduces the output length from m to $m^{1/3}$, which is still $n^{\Omega(1)}$. We can increase the output length by using a standard boosting technique as that developed by Nisan and Zuckerman [32, 40]. Specifically, we first use random bits to sample several blocks from the source, using a sampler in AC^0. We then apply our AC^0 extractor on the blocks backwards, and use the output of one block as the seed to extract from the previous block. When doing this we divide the seed into blocks each with the same length as the seed of the AC^0 extractor, apply the AC^0 extractor using each block as the seed, and then concatenate the outputs. This way each time the output will increase by a factor of $n^{\Omega(1)}$. Thus after a constant number of times it will become say $\Omega(k)$. Since each step is computable in AC^0, the whole construction is still in AC^0.

4 Explicit AC^0 extractors for bit-fixing sources

Our explicit AC^0 extractors for (oblivious) bit-fixing sources follow the high-level idea in [15]. Specifically, we first reduce the oblivious bit-fixing source to a non-oblivious bit-fixing source, and then apply an extractor for non-oblivious bit-fixing sources. This approach is natural in the sense that the best known extractors for oblivious bit-fixing sources (e.g., parity or [24]) can both work for small entropy and achieve very small error. Thus by the negative results in [15] and our paper, none of these can be in AC^0. However, extractors for non-oblivious bit-fixing sources are equivalent to resilient functions, and there are well known resilient functions in AC^0 such as the Ajtai-Linial function [1].

The construction in [15] is not explicit, but only existential for two reasons. First, at that time the Ajtai-Linial function is a random function, and there was no explicit construction matching it. Second, the conversion from oblivious-bit fixing source to non-oblivious bit-fixing source in [15] is to multiply the source by a random matrix, for which the authors of [15] showed its existence but were not able to give an explicit construction. Now, the first obstacle is solved by recent explicit constructions of resilient functions in AC^0 that essentially match the Ajtai-Linial function ([9, 28, 26]). Here we use the extractor in [26] that can output many bits. For the second obstacle, we notice that the extractors for non-oblivious bit-fixing sources in [9, 26] do not need the uniform bits to be independent, but rather only require $\mathsf{poly}(\log N)$-wise independence if N is the length of the source.

By exploiting this property, we can give an explicit construction of the matrix used to transfrom the original oblivious bit-fixing source. Our construction is natural and simpler than that in [15], in the sense that it is a matrix over F_2 while the matrix in [15] uses fields of larger size. Specifically, we will take a seeded extractor and view it as a bipartite graph with $N = n^{O(1)}$ vertices on the left, n vertices on the right and left degree $d = \mathsf{poly}(\log N) = \mathsf{poly}(\log n)$. We identify the right vertices with the n bits of the bit-fixing source, and for each left vertex we obtain a bit which is the parity of its neighbors. The new non-oblivious bit-fixing source is the N bit source obtained by concatenating the left bits.

Now suppose the original source has entropy $k = \delta n$ for some $\delta \geq 1/\mathsf{poly}(\log n)$, and let T denote the unfixed bits. A standard property of the seeded extractor implies that most of the left vertices have a good fraction of neighbors in T (i.e., an extractor is a good

sampler), so that each left bit obtained from these vertices is uniform. Next we would like to argue that they are $\mathsf{poly}(\log N)$-wise independent. For this we require the seeded extractor to have a stronger property: that it is a *design extractor* as defined by Li [25]. Besides being an extractor itself, a design extractor requires that any pair of left vertices have a small intersection of neighbors. Assuming this property, it is easy to show that if we take any small subset S of the "good" left vertices, then there is a bit in T that is only connected to a single vertex in S (i.e., a unique neighbor). Thus the XOR of any small enough subset of the "good" left bits is uniform, which indicates that they are some t-wise independent. Several explicit constructions of design extractors were given in [25], and for our applications it suffices to use a simple greedy construction. By adjusting the parameters, we can ensure that $t = \mathsf{poly}(\log N)$ which is enough for applying the extractor in [26]. In addition, the degree $d = \mathsf{poly}(\log N)$ so the parity of d bits can be computed in AC^0.

Once we have the basic extractor, we can use the same techniques as in [15] to reduce the error, and use the techniques by Gabizon et al.[13] to increase the output length (this is also done in [15]). Note that the techniques in [13] require a seeded extractor. In order for the whole construction to be in AC^0, we use our previously constructed seeded extractor in AC^0 which can output $(1-\gamma)k$ bits. Thus we obtain almost optimal explicit AC^0 extractors for oblivious bit-fixing sources. In contrast, the seeded extractor used in [15] only outputs $k/\mathsf{poly}(\log n)$ bits, and thus their (non-explicit) AC^0 extractor for oblivious bit-fixing sources also only outputs $k/\mathsf{poly}(\log n)$ bits.

5 Extractors with small locality for low entropy

Our basic extractor (Theorem 3) also enjoys the property of small locality, but it only works for large entropy. To get constructions for small min-entropy, we adapt the techniques in [6]. There the authors constructed a strong extractor family with small sparsity by randomly sampling an $m \times n$ matrix M and outputting MX, where X is the (n,k)-source. Each entry in M is independently sampled according to a Bernoulli distribution, and thus the family size is 2^{nm}. We derandomize this construction by sampling the second row to the last row using a random walk on an expander graph, starting from the first row. For the first row, we observe that the process of generating the entries and doing inner product with X can be realized by read-once small space computation, thus we can sample the first row using the output of a pseudorandom generator for space bounded computation (e.g., Nisan's generator [30]). We show that this gives us a very good condenser with small locality, i.e., Theorem 8. Combining the condenser with our previous extractors we then obtain strong extractor families with small locality.

6 Applications to pseudorandom generators

For cryptographic pseudorandom generators, we mainly adapt the approach of Applebaum [2], to the AC^0 setting. The construction of cryptographic pseudorandom generator *families* in [2] is based on random local functions. Specifically, given a random bipartite graph with n left vertices, m right vertices and right degree d (think of d as a constant), and a suitable predicate P on d bits, Applebaum showed that based on a conjecture on random local one-way functions, the m output bits obtained by applying P to the m subsets of input bits corresponding to the hyper edges give a distribution with high pseudo Shannon entropy. He then showed how to boost the output to have high pseudo min-entropy by concatenating several independent copies. At this point he used an extractor in NC^0 to turn the output into a pseudorandom string.

However, an extractor in NC^0 needs to have a large seed length (i.e., $\Omega(n)$), thus the NC^0 PRG constructed using this approach only achieves linear stretch. Another issue is that the NC^0 PRG is actually a collection of functions rather than a single function, because the random bits used to sample the bipartite graph is larger than the output length, and is treated as a public index to the collection of functions.

Here, by replacing the extractor with our AC^0 extractor we can achieve a polynomial stretch PRG (based on appropriate assumptions as in [2]), although now the PRG is in AC^0 instead of NC^0. In addition, we can get a single PRG instead of a collection of PRG functions, by including the random bits used to sample the bipartite graph as part of the seed. Since in the graph each right vertex only has a constant number d of neighbors, the sampling uses $md \log n$ bits and can be done in AC^0. To ensure that the PRG has a stretch, we take the sampled graph G and apply the *same* graph to several independent copies of n bit input strings. We show that we can still use the method in [2] to argue that this gives a a distribution with high pseudo Shannon entropy. We then use the same method as in [2] to turn it into a distribution with high pseudo min-entropy, and finally we apply our AC^0 extractor. This way we ensure that the $md \log n$ bits used to sample the graph G are "absorbed" by the stretch of the PRG, and thus we get a standard PRG instead of a collection of PRG functions.

For PRGs for space bounded computation, we simply adapt the PRG by Nisan and Zuckerman [32], which stretches $O(S)$ random bits to any $\mathsf{poly}(S)$ bits that fool space S computation. We now replace the seeded extractor used there by our AC^0 extractor. Notice that the Nisan-Zuckerman PRG simply applies the seeded extractor iteratively for a constant number of times, so the whole construction is still in AC^0.

7 Open problems

Our work leaves many natural open problems. First, in terms of the seed length and output length, our AC^0 extractor is only optimal when $k = \Omega(n)$. Is it possible to simultaneously achieve optimal seed length and output length when $k = n/\mathsf{poly}(\log n)$? Second, can we construct good AC^0 extractors for other classes of sources, such as independent sources and affine sources?

Turning to strong extractor families with small locality, again the parameters of our constructions do not match the parameters of optimal seeded extractors. In particular, our seed length is still $O(k)$ when the min-entropy k is small. Can we reduce the seed length further? We note that using our analysis together with the IW-generator/extractor, one can get something meaningful (i.e., a strong extractor family with a relatively short seed and small locality) even when $k = n^\alpha$ for some $\alpha > 1/2$. But it's unclear how to get below this entropy.

For pseudorandom generators in AC^0, there are also many interesting open problems left. For example, can we construct better cryptographic PRGs, or use weaker computational assumptions? In particular, it would be nice to construct a cryptographic PRG with polynomial stretch based on the one-wayness of a random local function with $m = (1 + \alpha)n$ instead of $m = n^{1+\alpha}$ as in our current construction. For space bounded computation, is it possible to match the exponentially small error of the Nisan-Zukerman PRG? Taking one step further, is it possible to construct PRGs in AC^0 for space bounded computation, with stretch matching the PRGs of Nisan [30] and Impagliazzo-Nisan-Wigderson [20]?

References

1. M. Ajtai and N. Linial. The influence of large coalitions. *Combinatorica*, 13(2):129–145, 1993.
2. Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. *SIAM Journal on Computing*, 42(5):2008–2037, 2013.
3. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in nc 0. *SIAM Journal on Computing*, 2006.
4. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. On pseudorandom generators with linear stretch in nc0. *Computational Complexity*, 17(1):38–69, 2008.
5. L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. Bpp has subexponential simulation unless Exptime has publishable proofs. *Computational Complexity*, 3:307–318, 1993.
6. Andrej Bogdanov and Siyao Guo. Sparse extractor families for all the entropy. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 553–560. ACM, 2013.
7. Ran Canetti, Yevgeniy Dodis, Shai Halevi, Eyal Kushilevitz, and Amit Sahai. Exposure-resilient functions and all-or-nothing transforms. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 453–469. Springer-Verlag, 2000.
8. J. L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.
9. Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing*, 2016.
10. Kuan Cheng and Xin Li. Randomness extraction in ac0 and with small locality. *arXiv preprint arXiv:1602.01530*, 2016.
11. Anindya De and Luca Trevisan. Extractors using hardness amplification. In *RANDOM 2009, 13th International Workshop on Randomization and Approximation Techniques in Computer Science*, 2009.
12. Stefan Dziembowski and Ueli Maurer. Tight security proofs for the bounded-storage model. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 341–350. ACM, 2002.
13. Ariel Gabizon, Ran Raz, and Ronen Shaltiel. Deterministic extractors for bit-fixing sources by obtaining an independent seed. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 394–403, 2004.
14. Oded Goldreich. Candidate one-way functions based on expander graphs. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 76–87. Springer, 2011.
15. Oded Goldreich, Emanuele Viola, and Avi Wigderson. On randomness extraction in ac0. In *30th Conference on Computational Complexity (CCC 2015)*, volume 33, pages 601–668. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015.
16. M. Grohe and N. Schweikardt. Lower bounds for sorting with few random accesses to external memory. In *Symposium on Principles of Database Systems (PODS)*, pages 238–249, 2005.
17. Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes. *Journal of the ACM*, 56(4), 2009.
18. J. Håstad. Almost optimal lower bounds for small depth circuits. In S. Micali, editor, *Randomness and Computation*, volume 5, pages 143–170. JAI Press, 1989.
19. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Construction of a pseudo-random generator from any one-way function. *SIAM Journal on Computing*, 28:12–24, 1993.

20 R. Impagliazzo, N. Nisan, and A. Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 356–364, 1994.

21 R. Impagliazzo and A. Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 220–229, 1997.

22 Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 538–545. IEEE, 1995.

23 Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of cryptology*, 9(4):199–216, 1996.

24 Jesse Kamp and David Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. *SIAM Journal on Computing*, 36(5):1231–1247, 2007.

25 Xin Li. Design extractors, non-malleable condensers and privacy amplification. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, pages 837–854, 2012.

26 Xin Li. Improved two-source extractors, and affine extractors for polylogarithmic entropy. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science*, 2016.

27 Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. *Journal of the ACM*, pages 607–620, 1993.

28 Raghu Meka. Explicit resilient functions matching Ajtai-Linial. *CoRR*, abs/1509.00092, 2015. `arXiv:1509.00092`.

29 Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On ε-biased generators in nc0. *Random Structures & Algorithms*, 29(1):56–81, 2006.

30 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12:449–461, 1992.

31 Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

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

33 Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

34 Periklis A Papakonstantinou, David P Woodruff, and Guang Yang. True randomness from big data. *Scientific reports*, 6, 2016.

35 R. Raz, O. Reingold, and S. Vadhan. Error reduction for extractors. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 191–201, 1999.

36 Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, pages 860–879, 2001.

37 S. Vadhan. On constructing locally computable extractors and cryptosystems in the bounded storage model. In *Advances in Cryptology — CRYPTO '03, 23rd Annual International Cryptology Conference, Proceedings*, 2003.

38 Emanuele Viola. The complexity of constructing pseudorandom generators from hard functions. *computational complexity*, 13(3-4):147–188, 2005.

39 Emanuele Viola. On constructing parallel pseudorandom generators from one-way functions. In *Computational Complexity, 2005. Proceedings. Twentieth Annual IEEE Conference on*, pages 183–197. IEEE, 2005.

40 D. Zuckerman. Randomness-optimal oblivious sampling. *Random Structures and Algorithms*, 11:345–367, 1997.

A Lower bounds for Error parameters of AC^0 extractors

Our negative results about the error of AC^0 extractors follow by a simple application of Fourier analysis and the well known spectrum concentration theorem of AC^0 functions [27].

▶ **Theorem 16** (LMN Theorem [27] [33]). *Let $f : \{-1,1\}^n \to \{-1,1\}$ be computable by AC^0 circuits of size $s > 1$ and depth $\mathtt{dth}$. Let $\epsilon \in (0, 1/2]$. There exists $t = O(\log(s/\epsilon))^{\mathtt{dth}-1} \cdot \log(1/\epsilon)$ s.t.*

$$\sum_{S\subseteq[n],|S|>t} \hat{f}_S^2 \leq \epsilon.$$

Now we prove Theorem 2.

Proof of Theorem 2. Without loss of generality, let $m = 1$. Let's transform the function space of Ext to $\{-1,1\}^{n+d} \to \{-1,1\}$, achieving function f. Let $\epsilon_0 = 1/2$. By Theorem 16, there exists $t = O(\log(s/\epsilon_0))^{\mathtt{dth}-1} \cdot \log(1/\epsilon_0) = O(\log s)^{\mathtt{dth}-1}$ s.t.

$$\sum_{S\subseteq[n+d],|S|\leq t} \hat{f}_S^2 > 1 - \epsilon_0 = 1/2.$$

Fix an $S = S_1 \cup \{i + n \mid i \in S_2\}$ with $|S| \leq t$, where $S_1 \subseteq [n], S_2 \subseteq \{1, 2, \ldots, d\}$. We know that

$$\hat{f}_S = \langle f, \chi_S \rangle = 1 - 2\Pr_u[f(u) \neq \chi_S(u)]$$

where u is uniformly drawn from $\{-1,1\}^{n+d}$.

For $a \in \{-1,1\}$, let X_a be the uniform distribution over $\{-1,1\}^n$ conditioned on $\prod_{i\in S_1} X_i = a$. Also for $b \in \{-1,1\}$, let R_b be the uniform distribution over $\{-1,1\}^d$ conditioned on $\prod_{i\in S_2} R_i = b$. So $\chi_S(x \circ r) = ab$ for $x \in \mathsf{supp}(X_a), r \in \mathsf{supp}(R_b)$. For special situations, saying $S_1 = \emptyset$ (or $S_2 = \emptyset$), let X_a (or R_b) be uniform.

As Ext is a strong (k, ϵ)-extractor, R_b only blows up the error by 2. Also note that by definition, X_a has entropy $n - 1$. So

$$\mathsf{dist}(f(X_a \circ R_b), U) \leq 2\epsilon,$$

where U is uniform over $\{-1,1\}$.

So

$$\forall a, b \in \{-1,1\}, |\Pr[f(X_a \circ R_b) \neq ab] - 1/2| \leq 2\epsilon.$$

Thus

$$\begin{aligned} |\Pr_u[f(u) \neq \chi_S(u)] - 1/2| &= |\sum_{a\in\{-1,1\}} \sum_{b\in\{-1,1\}} \frac{1}{4}(\Pr[f(X_a \circ R_b) \neq ab] - 1/2)| \\ &\leq \sum_{a\in\{-1,1\}} \sum_{b\in\{-1,1\}} \frac{1}{4}|\Pr[f(X_a \circ R_b) \neq ab] - 1/2| \\ &\leq 2\epsilon. \end{aligned} \tag{1}$$

Hence

$$|\hat{f}_S| = |1 - 2\Pr_u[f(u) \neq \chi_S(u)]| = 2|\Pr_u[f(u) \neq \chi_S(u)] - 1/2| \leq 4\epsilon.$$

As a result,

$$1/2 \leq \sum_{S\subseteq[n+d],|S|\leq t} \hat{f}_S^2 \leq \sum_{i=0}^{t} \binom{n+d}{i} (4\epsilon)^2.$$

So

$$\epsilon \geq \sqrt{\frac{1}{32\sum_{i=0}^{t}\binom{n+d}{i}}}.$$

As $\sum_{i=0}^{t}\binom{n+d}{i} \leq (\frac{e(n+d)}{t})^t = 2^{O(t\log(n+d))} = 2^{O(\log s)^{\mathtt{dth}-1}\log(n+d)}$, $\epsilon = 2^{-O(\log s)^{\mathtt{dth}-1}\log(n+d)}$.

The second assertion follows from a similar argument which is deferred to the full version [10]. ◀

B The basic construction of extractors in AC^0

Our basic construction is based on the general idea of I-W generator [21]. In [36], Trevisan showed that I-W generator is an extractor if we regard the string x drawn from the input (n,k)-source X as the truth table of a function f_x s.t. $f_x(\langle i\rangle), i \in [n]$ outputs the ith bit of x.

The construction of I-W generator involves a process of hardness amplifications from a worst-case hard function to an average-case hard function. There are mainly 3 amplification steps. Viola [38] summarizes these results in details, and we review them again. The first step is established by Babai et al. [5], which is an amplification from worst-case hardness to mildly average-case hardness.

▶ **Definition 17.** A boolean function $f : \{0,1\}^l \to \{0,1\}$ is δ-hard on uniform distributions for circuit size g, if for any circuit C with at most g gates ($\mathsf{size}(C) \leq g$), we have $\Pr_{x\leftarrow U}[C(x) = f(x)] < 1-\delta$.

▶ **Lemma 18** ([5]). *If there is a boolean function $f : \{0,1\}^l \to \{0,1\}$ which is 0-hard for circuit size $g = 2^{\Omega(l)}$ then there is a boolean function $f' : \{0,1\}^{\Theta(l)} \to \{0,1\}$ that is $1/\mathsf{poly}(l)$-hard for circuit size $g' = 2^{\Omega(l)}$.*

The second step is an amplification from mildly average-case hardness to constant average-case hardness, established by Impagliazzo [22].

▶ **Lemma 19** ([22]).

1. *If there is a boolean function $f : \{0,1\}^l \to \{0,1\}$ that is δ-hard for circuit size g where $\delta < 1/(16l)$, then there is a boolean function $f' : \{0,1\}^{3l} \to \{0,1\}$ that is $0.05\delta l$-hard for circuit size $g' = \delta^{O(1)}l^{-O(1)}g$.*

   $$f'(s,r) = \langle s, f(a_1)\circ f(a_2)\circ\cdots\circ f(a_l)\rangle$$

   Here $|s| = l$, $|r| = 2l$ and $|a_i| = l, \forall i \in [l]$. Regarding r as a uniform random string, $a_1,\ldots,a_l$ are generated as pairwise independent random strings from the seed r.
2. *If there is a boolean function $f : \{0,1\}^l \to \{0,1\}$ that is δ-hard for circuit size g where $\delta < 1$ is a constant, then there is a boolean function $f' : \{0,1\}^{3l} \to \{0,1\}$ that is $1/2 - O(l^{-2/3})$-hard for circuit size $g' = l^{-O(1)}g$, where*

   $$f'(s,r) = \langle s, f(a_1)\circ f(a_2)\circ\cdots\circ f(a_l)\rangle.$$

   Here $|s| = l$, $|r| = 2l$ and $|a_i| = l, \forall i \in [l]$. Regarding r as a uniform random string, $a_1,\ldots,a_l$ are generated as pairwise independent random strings from the seed r.

The first part of this lemma can be applied for a constant number of times to get a function having constant average-case hardness. After that the second part is usually applied for only once to get a function with constant average-case hardness such that the constant is large enough (at least 1/3).

The third step is an amplification from constant average-case hardness to even stronger average-case hardness, developed by Impagliazzo and Widgerson [21]. Their construction uses the following Nisan-Widgerson Generator [31] which is widely used in hardness amplification.

▶ **Definition 20** ((n,m,k,l)-design and Nisan-Widgerson Generator [31]). A system of sets $S_1, S_2, \ldots, S_m \subseteq [n]$ is an (n,m,k,l)-design, if $\forall i \in [m], |S_i| = l$ and $\forall i,j \in [m], i \neq j, |S_i \cap S_j| \leq k$.

Let $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ be an (n,m,k,l) design and $f : \{0,1\}^l \to \{0,1\}$ be a boolean function. The Nisan-Widgerson Generator is defined as $\mathsf{NW}_{f,\mathcal{S}}(u) = f(u|_{S_1}) \circ f(u|_{S_2}) \circ \cdots \circ f(u|_{S_m})$. Here $u|_{S_i} = u_{i_1} \circ u_{i_2} \circ \cdots \circ u_{i_m}$ assuming $S_i = \{i_1, \ldots, i_m\}$.

Nisan and Widgeson [31] showed that the (n,m,k,l)-design can be constructed efficiently.

▶ **Lemma 21** (Implicit in [31]). *For any $\alpha \in (0,1)$, for any large enough $l \in \mathbb{N}$, for any $m < \exp\{\frac{\alpha l}{4}\}$, there exists an $(n,m,\alpha l,l)$-design where $n = \lfloor \frac{10l}{\alpha} \rfloor$. This design can be computed in time polynomial of 2^n.*

The following is the third step of hardness amplification.

▶ **Lemma 22** (Implicit in [21]). *For any $\gamma \in (0, 1/30)$, if there is a boolean function $f : \{0,1\}^l \to \{0,1\}$ that is $1/3$-hard for circuit size $g = 2^{\gamma l}$, then there is a boolean function $f' : \{0,1\}^{l'=\Theta(l)} \to \{0,1\}$ that is $(1/2-\epsilon)$-hard for circuit size $g' = \Theta(g^{1/4}\epsilon^2 l^{-2})$ where $\epsilon \geq (500l)^{1/3} g^{-1/12}$.*

$$f'(a, s, v_1, w) = \langle s, f(a|_{S_1} \oplus v_1) \circ f(a|_{S_2} \oplus v_2) \circ \cdots f(a|_{S_l} \oplus v_l) \rangle$$

Here $(S_1, \ldots, S_l)$ is an $(|a|, l, \gamma l/4, l)$-design where $|a| = \lfloor \frac{40l}{\gamma} \rfloor$. The vectors $v_1, \ldots, v_l$ are obtained by a random walk on an expander graph, starting at v_1 and walking according to w where $|v_1| = l, |w| = \Theta(l)$. The length of s is l. So $l' = |a| + |s| + |v_1| + |w| = \Theta(l)$.

Our basic construction is an adjustment of the IW-Extractor.

▶ **Construction 23.** *For any $c_2 \in \mathbb{N}^+$ such that $c_2 \geq 2$ and any $k = \Theta(n/\log^{c_2-2} n)$, let X be an (n,k)-source . We construct a strong $(k, 2\epsilon)$ extractor $\mathsf{Ext}_0 : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ where $\epsilon = 1/n^\beta$, $\beta = 1/600$, $d = O(\log n)$, $m = k^{\Theta(1)}$. Let U be the uniform distribution of length d.*

1. *Draw x from X and u from U. Let $f_1 : \{0,1\}^{l_1} \to \{0,1\}$ be a boolean function such that $\forall i \in [2^{l_1}]$, $f_1(\langle i \rangle) = x_i$ where $l_1 = \log n$.*
2. *Run amplification step of Lemma 19 part 1 for c_2 times and run amplification step of Lemma 19 part 2 once to get function $f_2 : \{0,1\}^{l_2} \to \{0,1\}$ from f_1 where $l_2 = 3^{c_2+1} l_1 = \Theta(\log n)$.*
3. *Run amplification step Lemma 22 to get function $f_3 : \{0,1\}^{l_3} \to \{0,1\}$ from f_2 where $l_3 = \Theta(\log n)$.*
4. *Construct function Ext_0 such that $\mathsf{Ext}_0(x,u) = \mathsf{NW}_{f_3,\mathcal{S}}(u)$.*

Here $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ is a $(d, m, \theta l_3, l_3)$-design with $\theta = l_1/(900 l_3)$, $d = \lfloor 10 l_3/\theta \rfloor$, $m = \lfloor 2^{\frac{\theta l_3}{4}} \rfloor = \lfloor n^{\frac{1}{3600}} \rfloor$.

▶ **Lemma 24.** *In Construction 23, Ext_0 is a strong $(k, 2\epsilon)$ extractor.*

The proof follows from the "Bad Set" argument given by Trevisan [36]. In Trevisan [36] the argument is not explicit for strong extractors. Here our argument is explicit for proving that our construction gives a strong extractor.

Proof. We will prove that for every (n,k)-source X and for every $A : \{0,1\}^{d+m} \to \{0,1\}$ the following holds.

$$|\Pr[A(U_s \circ \mathsf{Ext}_0(X,U_s)) = 1] - \Pr[A(U) = 1]| \leq 2\epsilon$$

Here U_s is the uniform distribution over $\{0,1\}^d$ and U is the uniform distribution over $\{0,1\}^{d+m}$.

For every flat (n,k)-source X, and for every (fixed) function A, let's focus on a set $B \subseteq \{0,1\}^n$ such that $\forall x \in \mathsf{supp}(X)$, if $x \in B$, then

$$|\Pr[A(U_s \circ \mathsf{Ext}_0(x,U_s)) = 1] - \Pr[A(U) = 1]| > \epsilon.$$

According to Nisan and Widgerson [31], we have the following lemma.

▶ **Lemma 25** (Implicit in [31] [36]). *If there exists an A-gate such that*

$$|\Pr[A(U_s \circ \mathsf{Ext}_0(x,U_s)) = 1] - \Pr[A(U) = 1]| > \epsilon,$$

then there is a circuit C_3 of size $O(2^{\theta l_3} m)$, using A-gates, that can compute f_3 correctly for $1/2 + \epsilon/m$ fraction of inputs.

Here A-gate is a special gate that can compute the function A.

By Lemma 25, there is a circuit C_3 of size $O(m2^{\theta l_3}) = O(2^{\frac{5\theta l_3}{4}}) = O(n^{1/720})$, using A-gates, that can compute f_3 correctly for $1/2 + \epsilon/m \geq 1/2 + 1/n^{1/400}$ fraction of inputs.

By Lemma 22, there is a circuit C_2, with A-gates, of size at most $\Theta(n^{\frac{1}{30}})$ which can compute f_2 correctly for at least $2/3$ fraction of inputs.

According to Lemma 19 and our settings, there is a circuit C_1, with A-gates, of size $n^{\frac{1}{30}}\mathsf{poly}\log n$ which can compute f_1 correctly for at least $1 - 1/(c_1 \log^{c_2} n)$ fraction of inputs for some constant $c_1 > 0$.

Next we give an upper bound on the size of B. $\forall x \in B$, assume we have a circuit of size $S = n^{1/30}\mathsf{poly}(\log n)$, using A-gates, that can compute at least $1 - 1/(c_1 \log^{c_2} n)$ fraction of bits of x. The total number of circuits, with A-gates, of size S is at most $2^{\Theta(mS \log S)} = 2^{n^{1/15}\mathsf{poly}(\log n)}$, as A is fixed and has fan-in $m + d = O(m)$. Each one of them corresponds to at most $\sum_{i=0}^{n/(c_1 \log^{c_2} n)} \binom{n}{i} \leq (e \cdot c_1 \log^{c_2} n)^{n/(c_1 \log^{c_2} n)} = 2^{O(n/\log^{c_2-1} n)}$ number of x. So

$$|B| \leq 2^{n^{1/15}\mathsf{poly}(\log n)} 2^{O(n/\log^{c_2-1} n)} = 2^{O(n/(\log^{c_2-1} n)}.$$

As X is an (n,k)-source with $k = \Theta(n/\log^{c_2-2} n)$,

$$\Pr[X \in B] \leq |B| \cdot 2^{-k} \leq \epsilon.$$

Then we know,

$$\begin{aligned}
&|\Pr[A(U_s \circ \mathsf{Ext}(X,U_s)) = 1] - \Pr[A(U) = 1]| \\
=&\sum_{x \in B} \Pr[X = x] |\Pr[A(U_s \circ \mathsf{Ext}(x,U_s)) = 1] - \Pr[A(U) = 1]| \\
&+ \sum_{x \notin B} \Pr[X = x] |\Pr[A(U_s \circ \mathsf{Ext}(x,U_s)) = 1] - \Pr[A(U) = 1]| \\
\leq& 2\epsilon. \qquad (2)
\end{aligned}$$

◀

▶ **Lemma 26.** *The seed length of construction 23 is* $O(\log n)$.

Proof. We know that $l_1 = \log n, l_2 = 3^{c_2+1}l_1 = \Theta(\log n), l_3 = \Theta(\log n)$. Also $\mathcal{S}$ is a $(\lceil 10l_3/c \rceil = \Theta(l_3), m, cl_3, l_3)$-design. So $d = \lfloor 10l_3/c \rfloor = \Theta(l_3) = \Theta(\log n)$. ◀

▶ **Lemma 27.** *The function* Ext_0 *in Construction 23 is in* AC^0. *The circuit depth is* $c_2 + 5$. *The locality is* $\Theta(\log^{c_2+2} n) = \mathsf{poly}(\log n)$.

Please refer to the full version [10] for the proof of this lemma.

According to to Lemma 24, Lemma 26, Lemma 27, we have the following theorem.

▶ **Theorem 28** (The basic construction)**.** *For any* $c \in \mathbb{N}$, *any* $k = \Theta(n/\log^c n)$, *there exists an explicit strong* (k, ϵ)*-extractor* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *in* AC^0 *of depth* $c + 7$, *where* $\epsilon = n^{-1/600}$, $d = O(\log n)$, $m = \lfloor n^{\frac{1}{3600}} \rfloor$ *and the locality is* $\Theta(\log^{c+4} n) = \mathsf{poly}\log n$.