Electronic Colloquium on Computational Complexity, Revision 3 of Report No. 18 (2016)

Randomness Extraction in AC^0 and with Small Locality

Kuan Cheng * Xin Li †

November 4, 2016

Abstract

We study two variants of randomness extractors. The first one, as studied by Goldreich et al. [19], is extractors that can be computed by AC^0 circuits. The second one, as introduced by Bogdanov and Guo [8], is (strong) extractor families that consist of sparse transformations, i.e., functions that have a small number of overall input-output dependencies (called *sparse extractor families*). In this paper we focus on the stronger condition where any function in the family can be computed by local functions. The parameters here are the length of the source n, the min-entropy $k = k(n)$, the seed length $d = d(n)$, the output length $m = m(n)$, the error $\epsilon = \epsilon(n)$, and the locality of functions $\ell = \ell(n)$.

In the AC^0 extractor case, we study both seeded extractors and deterministic extractors for bit-fixing sources. Our negative results show that the error of such extractors cannot be better than $2^{-\mathsf{poly}(\log n)}$. Together with the lower bound on entropy in [19] this almost completely characterizes the power of AC^0 extractors. Our positive results substantially improve the positive results in [19], where for weak sources with $k \geq n/\mathsf{poly}(\log n)$ a seed length of $O(m)$ is required to extract m bits with error $1/\mathsf{poly}(n)$. We give constructions of strong seeded extractors for $k \geq n/\mathsf{poly}(\log n)$, with seed length $d = O(\log n)$, output length $m = (1-\gamma)k$ for any constant $0 < \gamma < 1$, and error any $1/\mathsf{poly}(n)$. In addition, we can reduce the error to $2^{-\mathsf{poly}(\log n)}$ at the price of increasing the seed length to $d = \mathsf{poly}(\log n)$, essentially matching our error bound. We give two applications of such extractors to the constructions of pseudorandom generators in AC^0 that are cryptographically secure, and that fool small space computation. In addition, we give the first *explicit* AC^0 extractor for oblivious bit-fixing sources with entropy $k \geq n/\mathsf{poly}(\log n)$, output length $m = (1-\gamma)k$ and error $2^{-\mathsf{poly}(\log n)}$, which are essentially optimal.

In the case of sparse extractor families, Bogdanov and Guo [8] gave constructions for any min-entropy k with locality at least $O(n/k \log(m/\epsilon) \log(n/m))$, but the family size is quite large, i.e., 2^{nm}. Equivalently, this means the seed length is at least nm. In this paper we significantly reduce the seed length. For $k \geq n/\mathsf{poly}(\log n)$ and $\epsilon \geq 2^{-k^{\Omega(1)}}$, we show how to get a strong seeded extractor with seed length $d = O(\log n + \frac{\log^2(1/\epsilon)}{\log n})$, output length $m = k^{\Omega(1)}$ and locality $\log^2(1/\epsilon)\mathsf{poly}(\log n)$. In addition, for min-entropy $k = \Omega(\log^2 n)$ and error $\epsilon \geq 2^{-k^{\Omega(1)}}$, we give a strong seeded extractor with seed length $d = O(k)$, $m = (1-\gamma)k$ and locality $\frac{n}{k}\log^2(1/\epsilon)(\log n)\mathsf{poly}(\log k)$. As an intermediate tool for this extractor, we construct a condenser that condenses an (n, k)-source into a $(10k, \Omega(k))$-source with seed length $d = O(k)$, error $2^{-\Omega(k)}$ and locality $\Theta(\frac{n}{k}\log n)$.

*kcheng17@jhu.edu. Department of Computer Science, Johns Hopkins University. Supported in part by NSF Grant CCF-1617713.

†lixints@cs.jhu.edu. Department of Computer Science, Johns Hopkins University. Supported in part by NSF Grant CCF-1617713.

ISSN 1433-8092

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.[1] 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 [38], 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 1.1 ([38])**.** *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., [20]).

While in general "explicit constructions" means constructions that 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 [11], and Trevisan's extractor [42], which can be computed by highly uniform constant-depth circuits of polynomial size with parity gates. Motivated by this, Goldreich et al. [19] studied the problem of constructing randomness extractors in AC^0 (i.e., constant depth circuits of polynomial size with unbounded fan-in gates). They considered both the question of constructing seeded randomness extractors in AC^0 and the question of constructing seedless (deterministic) extractors for bit-fixing sources. From the complexity aspect, this also helps us better understand the computational power of the class AC^0. We continue with their study in this paper.

In a similar flavor, one can also consider randomness extractors that can be computed by local functions, i.e., where every output bit only depends on a small number (say ℓ) of input bits. However, one can easily see that in this case, just fixing at most ℓ bits of the weak source will cause the extractor to fail (at least in the strong extractor case). To get around this, Bogdanov and Guo [8] introduced the notion of sparse extractor families. These are a family of functions such that each function has a small number of overall input-output dependencies, 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 need. As an example, the authors in [8] used such extractors to obtain a transformation of non-uniform one-way functions into non-uniform pseudorandom generators that preserve output locality. We recall the definition of such extractors in [8].

Definition 1.2. *[8] (sparse extractor family) An* extractor family *for* (n, k)*-sources with error* ϵ *is a distribution* H *on functions* $\{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ *such that for any* (n, k)*-source* X*, we have*

$$|(H, H(X, U_s)) - (H, U_m)| \leq \epsilon.$$

The extractor family is strong *if* $s = 0$*. Moreover, the family is* t*-sparse if for any function in the family, the number of input-output pairs* (i, j) *such that the* j*'th output bit depends on the* i*'th input bit is at most* t*. The family is* ℓ*-local if for any function in the family, any output bit depends on at most* ℓ *input bits.*

[1] 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}.

In this paper, we continue the study of such extractors under the stronger condition of the family being ℓ-local (instead of just being sparse). 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. Formally, we define such extractors as follows.

Definition 1.3. *A seeded extractor* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a strong* (k,ϵ)*-extractor family with locality* ℓ*, if* Ext *satisfies the following two conditions.*

- *For any* (n,k)*-source* X *and independent uniform seed* $R \in \{0,1\}^d$*, we have*

$$|(\mathsf{Ext}(X,R),R) - U_{m+d}| \leq \epsilon.$$

- *For any fixing of the seed* $R = r$*, we have that* $\mathsf{Ext}(x,r)$ *is computable by* ℓ*-local functions, i.e., each output bit depends on at most* ℓ *bits of* x*.*

It can be seen that our definition of extractor families with small locality is a stronger notion than sparse extractor families. Indeed, an extractor family with m output bits and locality ℓ is automatically an ℓm-sparse extractor family, while the other direction may not hold.

Comparing with t-local extractors in [43] It is worthwhile to compare our definition of a strong extractor family with small locality to the definition of t-local extractors by Vadhan [43]. For a t-local extractor, one requires that for any fixing of the seed r, the outputs of the function $\mathsf{Ext}(x,r)$ *as a whole* depend on only t bits of x. In contrast, in our definition we only require that *each output bit* of the function $\mathsf{Ext}(x,r)$ depends on at most ℓ bits of x, while as a whole the output bits can depend on more than ℓ bits of x.

Of course, the definition of t-local extractors is stronger than ours, since any t-local extractor also has locality at most ℓ according to our definition. However, the construction of t-local extractors in [43], 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. Furthermore, by a lower bound in [43], the parameter t in local-extractors is at least $\Omega(nm/k)$, which is larger than the output length m. Although this is inevitable for local extractors, we can construct strong extractor families with long output and small locality (i.e., $\ell \ll m$).

1.1 Prior Work and Resuls

As mentioned before, Goldreich et al. [19] studied the problem of constructing randomness extractors in AC^0. They showed that in the case of strong seeded extractor, even extracting a single bit is impossible if $k < n/\mathsf{poly}(\log n)$. When $k \geq n/\mathsf{poly}(\log n)$, they 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 in this case the seed length is longer than the output length.[2] In the non-strong extractor case, they showed that extracting $r + \Omega(r)$ bits 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 of the above positive results have error $1/\mathsf{poly}(n)$. Therefore, a natural and main open problem left in [19] is to see if one can construct randomness extractors in AC^0 with shorter seed and longer output. Specifically, [19] 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 [19] the authors conjectured that the answer is negative. Another drawback of the constructions in [19] is that they only achieve error $1/\mathsf{poly}(n)$, and it is another open question to see if one can do better.

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

Goldreich et al. [19] 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 [10, 30]. In this case, a standard application of Håstad's switching lemma [21] 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 [19] is then a theorem which shows that there *exist* 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 [19] did not give explicit constructions of such extractors.

We now turn to sparse extractor families. The authors in [8] 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 roughly corresponds to locality $O(n/m \log(m/\epsilon) \log(n/m)) \geq O(n/k \log(m/\epsilon) \log(n/m)) \geq O(n/k \log(n/\epsilon))$ whenever $k \leq n/2$. They also showed that such sparsity is necessary when $n^{0.99} \leq m \leq n/6$ and ϵ is a constant. However, the main drawback of the construction in [8] 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 (in fact, since the distribution H is not uniform, it will take even more random bits to sample from the family). Therefore, a main open problem left in [8] is to reduce the size of the family (or, equivalently, the seed length).

De and Trevisan [13], using similar techniques as ours, also obtained a strong extractor for (n, k) sources with $k = \delta n$ for any constant δ with seed length $d = O(\log n)$ 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. Their extractor outputs $k^{\Omega(1)}$ bits, but the error is only $n^{-\alpha}$ for some small constant $0 < \alpha < 1$. Our results apply to a much wider setting of parameters. Indeed, as we shall see in the following, we can handle min-entropy as small as $k = \Omega(\log^2 n)$ and error as small as $2^{-k^{\Omega(1)}}$.

1.2 Our Results

As in [19], in this paper we obtain both negative results and positive results about randomness extraction in AC^0. While the negative results in [19] 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 is quite large. Specifically, we have

Theorem 1.4. *(For general weak sources) Let* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^{l=\mathsf{poly}(n)} \to \{0,1\}^m$ *be a strong* $(k = n - 1, \epsilon)$*-extractor which can be computed by* AC^0 *circuits of depth* d*. Then* $\epsilon \geq 1/2^{O(\log^d n)}$.

(For bit-fixing sources) Let $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^{l=\mathsf{poly}(n)} \to \{0,1\}^m$ *be a strong extractor with error* ϵ*, for any* $(n, k = n - \mathsf{poly}\log n)$*-bit-fixing sources, which can be computed by* AC^0 *circuits of depth* d*. Then* $\epsilon \geq 1/2^{O(\log^d n)}$.

Thus, our results combined with the lower bounds on the entropy required in [19] 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 [19] 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 [19]. To start with, we have the following theorem.

Theorem 1.5. *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 [19]. 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 [8] (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 [13], 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 seed length of the extractor in [13] becomes $\mathsf{poly}(\log n)$.

Next, we can boost our construction to extract almost all the entropy. Specifically, we have

Theorem 1.6. *For any constant $c \in \mathbb{N}$, any $k = \delta n = \Omega(n/\log^c n)$, 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 $O(c) + O(1)$ with either of the following parameters.*

1. *$m = \Omega(\delta k)$ and $d = O(\log n)$.*
2. *$m = (1-\gamma)k$ for any constant $0 < \gamma < 1$ and $d = O(\frac{1}{\delta} \log n)$.*

Note that if δ is a constant, then we can extract $(1-\gamma)k$ bits with seed length $O(\log n)$ and error $\epsilon = 1/\mathsf{poly}(n)$, which is essentially optimal. In the case where $k = n/\mathsf{poly}(\log n)$, we can either use $O(\log n)$ bits to extract $k/\mathsf{poly}(\log n)$ bits or use $\mathsf{poly}(\log n)$ bits to extract $(1-\gamma)k$ bits.

By increasing the seed length, we can achieve even smaller error with extractors in AC^0, almost matching our lower bounds on error. Specifically, we have the following theorem.

Theorem 1.7. *For any constants $c_1, c_2 \in \mathbb{N}$, $\gamma \in (0,1)$, any $k = \Omega(n/\log^{c_1} n)$, and any $\epsilon = 2^{-\log^{c_2} n}$, 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 $O(c_1 + c_2) + O(1)$, where $d = \mathsf{poly}(\log n)$ and $m = (1-\gamma)k$.*

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 [19]. Further, the output length and error of our extractor are almost optimal, while the output length in [19] is only $k/\mathsf{poly}(\log n)$. Specifically, we have

Theorem 1.8. *For any constant $a, c \in \mathbb{N}$ and any constant $\gamma \in (0,1]$, there exists an explicit deterministic $(k = \Theta(n/\log^a n), \epsilon = 2^{-\log^c n})$-extractor $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^{(1-\gamma)k}$ that can be computed by AC^0 circuits of depth $\Theta(a+c)$, for any (n,k)-bit-fixing sources.*

Next, we discuss our results for sparse extractor families. Unfortunately, Theorem 1.6 and Theorem 1.7 do not preserve small locality as in Theorem 1.5, because our output length boosting step does not preserve locality. However, we can still reduce the error of Theorem 1.5 while keeping the locality small. Specifically, we have the following theorem.

Theorem 1.9. *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 $d = O(\log n + \frac{\log^2(1/\epsilon)}{\log n})$, $m = k^{\Theta(1)}$ and locality $\log^2(1/\epsilon)\mathsf{poly}(\log n)$.*

Finally, we consider 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 1.10. *A function* $\mathsf{Cond} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{n_1}$ *is a strong* $(n, k, n_1, k_1, \epsilon)$*-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 1.11. *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* $\Theta(\frac{n}{k} \log n)$*.*

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 1.12. *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 [8]. However, our locality becomes slightly worse, i.e., the dependence on ϵ changes from $\log(1/\epsilon)$ to $\log^2(1/\epsilon)$. Whether one can improve this is an interesting open problem.

1.3 Applications to pseudorandom genrators 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 for a certain class of distinguishers.

Definition 1.13. *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* $\mathcal{A} \in \mathcal{C}$*, we have that*

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

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., [23]), 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 [28] 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 [47] 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 [18].

In the case of unconditional PRGs, for $d \geq 5$ Mossel et al. [35] 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.

1.3.1 Our PRGs

In this paper 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 1.14 (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 1.15 (Goldreich's Random Local Function [18])**.** *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 1.16 (One-way Function for 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.

Theorem 1.17. *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.*
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 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 [38].

Theorem 1.18. *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 [38], our PRG is in 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 [38]. 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 $2^{-\mathsf{poly}(\log n)}$.

1.4 Overview of the Constructions and Techniques

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 by Linial, Mansour, and Nisan [33]. We present it in Section 3. We now briefly describe the constructions that give our positive results. To get strong randomness extractors in AC^0, 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 $\mathsf{poly}(n)$.

1.4.1 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 modification of the Impagliazzo-Widgerson pseudorandom generator [25], interpreted as a randomness extractor in the general framework found by Trevisan [42]. In the IW-generator, first one takes a Boolean function on n bits that is worst case hard for circuits of size $2^{\Omega(n)}$, and uses a series of hardness amplification steps to get another function that cannot be predicted with advantage more than $2^{-\Omega(n)}$ by circuits of size $2^{\Omega(n)}$. One can now use the Nisan-Widgerson generator [37] together with this new function to get a pseudorandom generator that stretches $O(\log n)$ random bits to n bits that fool any polynomial size circuit. Note that in the final step the hard function is applied on only $O(\log n)$ bits, so one can think of the initial Boolean function to be on $\log n$ bits.

Trevisan [42] 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 apply the IW-generator, then by setting parameters appropriately (e.g., set output length to be n^α) one gets an extractor. The reason is that if the function is not an extractor, then one can "reconstruct" part of the source X. More specifically, by the same argument of the IW-generator, one can show that any $x \in \mathsf{supp}(X)$ that makes the output of the extractor to fail a certain

statistical test T, can be computed by a small size circuit (when viewing x as the truth table of the function) with T gates. Since the total number of such circuits is small, the number of such bad elements in $\mathsf{supp}(X)$ is also small. This extractor can work for min-entropy $k \geq n^{\alpha}$.

However, this extractor itself is not in AC^0 (which should be no surprise 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. In more details, the construction has four steps, with the first three steps used for hardness amplification and the last step applying the NW-generator. In hardness amplification, the first step is developed by Babai et al. [6] to obtain a mild average-case hard function from a worst-case hard function; the second step involves a constant number of sub steps, with each sub step amplifying the hardness by using Impagliazzo's hard core set theorem [27], and eventually obtain a function with constant hardness; the third step is developed by Impagliazzo and Widgerson [25], which uses a derandomized direct-product generator to obtain a function that can only be predicted with exponentially small advantage. By carefully examining each step one can see that the only step not in AC^0 is actually the fist step of hardness amplification. 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 concatenate the outputs. All of these 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 [42]. 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 comptues* 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., $n^{-\beta}$. Note that in each hardness amplification step the output of f' only depends on $O(\log n)$ outputs from f, thus our extractor also enjoys the property of small locality, i.e., $\mathsf{poly}(\log n)$ since the construction only has a constant number of hardness amplification steps.

1.4.2 Error reduction

We now describe how we reduce the error of the extractor. We will use techniques similar to that of Raz et al. [39], in which the authors showed a general way to reduce the error of strong seeded extractors. However, the reduction in Raz et al. [39] does not preserve the AC^0 property or small locality, thus we cannot directly use it. Nevertheless, we will still use a lemma from [39], 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 $\mathsf{Ext}(x, r)$.

Suppose now we want to achieve an error of any $1/\mathsf{poly}(n)$. 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 $1/\mathsf{poly}(n)$.

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. [19].

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 $2^{-\mathsf{poly}(\log n)}$ and the construction is till in AC^0; although we need to change Ext'' a little bit and the seed length now becomes $\mathsf{poly}(\log n)$. In addition, our error reduction step also preserves small locality.

1.4.3 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 [38, 49]. Specifically, we first use random bits to sample from the source for several times (using a sampler in AC^0), and the outputs will form a block source. We then apply our AC^0 extractor on the block source 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. Unfortunately, this step does not preserve small locality.

1.4.4 Explicit AC^0 extractors for bit-fixing source

Our explicit AC^0 extractors for (oblivious) bit-fixing sources follow the high-level idea in [19]. 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. We note that in general, there are much better extractors for oblivious bit-fixing sources than for non-oblivious bit-fixing sources, e.g., the simple parity function or the extractor by Kamp and Zuckerman [30]. However, these extractors can both work for small entropy and achieve very small error. Thus by the negative results in [19] and our paper, none of these can be in AC^0. For non-oblivious bit-fixing sources, however, extractors are equivalent to resilient functions, and there are well

known resilient functions in AC^0, such as the Ajtai-Linial function [1]. Thus, the approach of first reducing to non-oblivious bit-fixing source seems natural.

The construction in [19] 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 [19] is to multiply the source by a random matrix, for which the authors of [19] 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 ([12, 34, 32]). Here we use the extractor in [32] that can output many bits. For the second obstacle, we notice that the extractors for non-oblivious bit-fixing sources in [12, 32] 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 multiply the original oblivious bit-fixing source. Our construction is natural and simpler than that in [19], in the sense that it is a matrix over F_2 while the matrix in [19] 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 $k = \alpha n$ for some $\alpha \geq 1/\mathsf{poly}(\log n)$, 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 [31]. 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 [31], 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 [32]. In addition, the degee $d = \mathsf{poly}(\log N)$ 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 [19] to reduce the error, and use the techniques by Gabizon et al.[14] to increase the output length (this is also done in [19]). Note that the techniques in [14] 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 [19] 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.

1.4.5 Applications to pseudorandom generators

For cryptographic pseudorandom generators, we mainly adapt the approach of Applebaum [2], to the AC^0 setting. In [2], Applebaum constructed cryptographically secure pseudorandom generator *families* in NC^0. His construction 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 [38], 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.

1.4.6 Extractors with small locality for low entropy

To get strong extractor families with small locality for min-entropy $k = \Omega(\log^2 n)$, we adapt the techniques in [8]. 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 [36]). We show that this gives us a very good condenser with small locality, i.e., Theorem 1.11. Combining the condenser with our previous extractors we then obtain strong extractor families with small locality.

1.5 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 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. In addition, our technique to increase output length fails to preserve locality. Is it possible to develop a locality-preserving technique for output length optimization? Furthermore, in general the locality

in our construction is a little worse in terms of the error ϵ than that of [8] (i.e., $\log^2(1/\epsilon)$ vs. $\log(1/\epsilon)$). This stems from our error reduction technique. Can we improve it to reduce the locality? Finally, a basic question here is still not clear: what is the correct relation between the six parameters input length n, min-entropy k, seed length d, output length m, error ϵ, and locality ℓ? It would be nice to obtain a matching upper bound and lower bound, as in the case of standard seeded extractors. We conjecture that a lower bound of locality $\frac{n}{k}\log(n/\epsilon)$ should hold, although we were not able to prove it in general.

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 [36] and Impagliazzo-Nisan-Wigderson [24]?

1.6 Organization of this Paper

The rest of the paper is organized as follows. In Section 2 we review some basic definitions and the relevant background. In Section 4 we describe our construction of a basic extractor in AC^0, and with small locality. Section 5 describes the error reduction techniques for AC^0 extractors. In Section 6 we show how to increase the output length of AC^0 extractors. Section 7 deals with error reduction for extractor families with small locality. In Section 9 we give our condenser and extractor with small locality for low entropy sources. in Section 10 we present our applications to pseudorandom generators in AC^0. We put some omitted details in Appendix A.

2 Preliminaries

For any $i \in \mathbb{N}$, we use $\langle i \rangle$ to denote the binary string representing i. Let $\langle \cdot, \cdot \rangle$ denote the inner product of two binary strings having the same length. Let $|\cdot|$ denote the length of the input string. Let $w(\cdot)$ denote the weight of the input binary string. For any strings x_1 and x_2, let $x_1 \circ x_2$ denote the concatenation of x_1 and x_2. For any strings $x_1, x_2, \ldots, x_t$, let $\bigcirc_{i=1}^{t} x_i$ denote $x_1 \circ x_2 \circ \cdots \circ x_t$.

Let $\mathsf{supp}(\cdot)$ denote the support of the input random variable.

Definition 2.1 (Weak Random Source, Block Source)**.** *The min-entropy of a random variable X is*

$$H_\infty(X) = \min_{x \in \mathsf{supp}(X)} \{-\log \Pr(X = x)\}.$$

We say a random variable X is an (n,k)-source if the length of X is n and $H_\infty(X) \geq k$. We say $X = \bigcirc_{i=1}^{m} X_i$ is an $((n_1,k_1),(n_2,k_2),\ldots,(n_m,k_m))$-block source if $\forall i \in [m]$, $\forall x \in \mathsf{supp}(\bigcirc_{j=1}^{i-1} X_j)$, $X_i|_{\bigcirc_{j=1}^{i-1} X_j = x}$ is an (n_i,k_i)-source.

For simplicity, if $n_1, n_2, \cdots, n_m$ are clear from the context, then we simply say that the block source X is a $(k_1, k_2, \ldots, k_m)$-block source.

We say an (n,k)-source X is a flat (n,k)-source if $\forall a \in \mathsf{supp}(X)$, $\Pr[X = a] = 2^{-k}$. In this paper, X is usually a random binary string with finite length. So $\mathsf{supp}(X)$ includes all the binary strings of that length such that $\forall x \in \mathsf{supp}(X), \Pr[X = x] > 0$.

Bit-fixing source is a special kind of weak source. In this paper we also consider deterministic extractors for bit-fixing source.

Definition 2.2 (Non-oblivious Bit-Fixing Sources)**.** *A source X on $\{0,1\}^n$ is a (q,t,γ)-non-oblivious bit-fixing source (in short, NOBF source) if there exists a subset $Q \subseteq [n]$ of size at most q and a sequence of functions $f_1, f_2, \ldots, f_n : \{0,1\}^{|I|} \to \{0,1\}$ such that the joint distribution of the bits indexed by $\bar{Q} = [n] \backslash Q$ (denoted by $X_{\bar{Q}}$) is (t,γ)-wise independent (γ-close to a t-wise independent source) and $X_i = f_i(X_{\bar{Q}})$ for every $i \in Q$.*

Bit-fixing sources are special non-oblivious bit-fixing sources. An (n,t)-bit-fixing source is defined to be an $(n-t,t,0)$-non-oblivious bit-fixing source.

We use U to denote the uniform distribution. In the following, we do not always claim the length of U, but its length can be figured out from the context.

Definition 2.3 (Statistical Distance)**.** *The statistical distance between two random variables X and Y, where $|X| = |Y|$, is $\mathsf{SD}(X,Y)$ which is defined as follows.*

$$\mathsf{SD}(X,Y) = 1/2 \sum_{a \in \{0,1\}^{|X|}} |\Pr[X=a] - \Pr[Y=a]|$$

Lemma 2.4 (Properties of Statistical Distance [5])**.** *Statistical distance has the following properties.*

1. *(Triangle Inequality) For any random variables X, Y, Z, such that $|X| = |Y| = |Z|$, we have*

$$\mathsf{SD}(X,Y) \leq \mathsf{SD}(X,Z) + \mathsf{SD}(Y,Z).$$

2. *For any $n, m \in \mathbb{N}^+$, any deterministic function $f : \{0,1\}^n \to \{0,1\}^m$ and any random variables X, Y over $\{0,1\}^n$, $\mathsf{SD}(f(X), f(Y)) \leq \mathsf{SD}(X,Y)$.*

Definition 2.5 (Extractor)**.** *A (k,ϵ)-extractor is a function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with the following property. For every (n,k)-source X, the distribution $\mathsf{Ext}(X,U)$ is within statistical distance ϵ from uniform distributions over $\{0,1\}^m$.*

A strong (k,ϵ)-extractor is a function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with the following property. For every (n,k)-source X, the distribution $U \circ \mathsf{Ext}(X,U)$ is within statistical distance ϵ from uniform distributions over $\{0,1\}^{d+m}$. The entropy loss of the extractor is $k-m$.

The existence of extractors can be proved using the probabilistic method. The result is stated as follows.

Theorem 2.6 ([45])**.** *For any $n, k \in \mathbb{N}$ and $\epsilon > 0$, there exists a strong (k,ϵ)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ such that $d = \log(n-k) + 2\log(1/\epsilon) + O(1), m = k - 2\log 1/\epsilon + O(1)$.*

In addition, researchers have found explicit extractors with almost optimal parameters, for example we have the following theorem.

Theorem 2.7 ([20])**.** *For every constant $\alpha > 0$, every $n, k \in \mathbb{N}$ and $\epsilon > 0$, there exist an explicit construction of strong (k,ϵ)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with $d = O(\log \frac{n}{\epsilon}), m \geq (1-\alpha)k$.*

We also use the following version of Trevisan's extractor [42].

Theorem 2.8 (Trevisan's Extractor [42])**.** *For any constant $\gamma \in (0,1]$, let $k = n^\gamma$. For any $\epsilon \in (0, 2^{-k/12})$, there exists an explicit construction of (k,ϵ)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ such that $d = O((\log n/\epsilon)^2/\log n)$, $m \in [36, k/2)$.*

For block sources, randomness extraction can be done in parallel, using the same seed for each block.

Lemma 2.9 (Block Source Extraction)**.** *For any $t \in \mathbb{N}^+$, let $X = \bigcirc_{i=1}^{t} X_i$ be any $(k_1, k_2, \ldots, k_t)$-block source where for each $i \in [t]$, $|X_i| = n_i$. For every $i \in [t]$, let $\mathsf{Ext}_i : \{0,1\}^{n_i} \times \{0,1\}^d \to \{0,1\}^{m_i}$ be a strong (k_i, ϵ_i)-extractor. Then the distribution $R \circ \mathsf{Ext}_1(X_1, R) \circ \mathsf{Ext}_2(X_2, R) \circ \ldots \circ \mathsf{Ext}_t(X_t, R)$ is $\sum_{i\in[t]} \epsilon_i$-close to uniform, where R is uniformly sampled from $\{0,1\}^d$, and independent of X.*

Proof. We use induction. If the source has only 1 block, then the statement is true by the definition of strong extractors.

Assume for $(t-1)$ blocks, the statement is true. We view $\mathsf{Ext}_1(X_1, R) \circ \mathsf{Ext}_2(X_2, R) \circ \ldots \circ \mathsf{Ext}_t(X_t, R)$ as $Y \circ \mathsf{Ext}_t(X_t, R)$. Here $Y = \mathsf{Ext}_1(X_1, R) \circ \mathsf{Ext}_2(X_2, R) \circ \ldots \circ \mathsf{Ext}_{t-1}(X_{t-1}, R)$. Let U_1, U_2 be two independent uniform distributions, where $|U_1| = |Y| = m$ and $|U_2| = m_t$. Then

$$\begin{aligned} &\mathsf{SD}(R \circ Y \circ \mathsf{Ext}_t(X_t, R), R \circ U_1 \circ U_2) \\ \leq &\mathsf{SD}(R \circ Y \circ \mathsf{Ext}_t(X_t, R), R \circ U_1 \circ Z) + \mathsf{SD}(R \circ U_1 \circ Z, R \circ U_1 \circ U_2). \end{aligned} \tag{1}$$

Here Z is the random variable such that $\forall r \in \{0,1\}^d, \forall y \in \{0,1\}^m$, $Z|_{R=r,U_1=y}$ has the same distribution as $\mathsf{Ext}_t(X_t, R)|_{R=r,Y=y}$.

First we give the upper bound of $\mathsf{SD}(R \circ Y \circ \mathsf{Ext}_t(X_t, R), R \circ U_1 \circ Z)$.

$$\begin{aligned} &\mathsf{SD}(R \circ Y \circ \mathsf{Ext}_t(X_t, R), R \circ U_1 \circ Z) \\ =&\frac{1}{2} \sum_{r\in\{0,1\}^d} \sum_{y\in\{0,1\}^m} \sum_{z\in\{0,1\}^{m_t}} |\Pr[R = r] \Pr[Y = y|_{R=r}] \Pr[\mathsf{Ext}_t(X_t, R) = z|_{R=r,Y=y}] \\ &- \Pr[R = r] \Pr[U_1 = y] \Pr[Z = z|_{R=r,U_1=y}]| \\ =&\frac{1}{2} \sum_{r\in\{0,1\}^d} \sum_{y\in\{0,1\}^m} \sum_{z\in\{0,1\}^{m_t}} \Pr[R = r] \Pr[Z = z|_{R=r,U_1=y}]| \Pr[Y = y|_{R=r}] - \Pr[U_1 = y]| \\ =&\frac{1}{2} \sum_{r\in\{0,1\}^d} \sum_{y\in\{0,1\}^m} \Pr[R = r]| \Pr[Y = y|_{R=r}] - \Pr[U_1 = y]| \sum_{z\in\{0,1\}^{m_t}} \Pr[Z = z|_{R=r,U_1=y}] \\ =&\frac{1}{2} \sum_{r\in\{0,1\}^d} \sum_{y\in\{0,1\}^m} \Pr[R = r]| \Pr[Y = y|_{R=r}] - \Pr[U_1 = y]| \\ =&\mathsf{SD}(R \circ Y, R \circ U) \\ \leq &\sum_{i=1}^{t-1} \epsilon_i. \end{aligned} \tag{2}$$

Next we give the upper bound of $\mathsf{SD}(R \circ U_1 \circ Z, R \circ U_1 \circ U_2)$.

$$\begin{aligned} &\mathsf{SD}(R \circ U_1 \circ Z, R \circ U_1 \circ U_2) \\ =&\frac{1}{2} \sum_{r\in\{0,1\}^r} \sum_{u\in\{0,1\}^m} \sum_{z\in\{0,1\}^{m_t}} |\Pr[R = r] \Pr[U_1 = u] \Pr[Z = z|_{R=r,U_1=u}] \\ &- \Pr[R = r] \Pr[U_1 = u] \Pr[U_2 = z]| \\ =&\frac{1}{2} \sum_{r\in\{0,1\}^r} \sum_{u\in\{0,1\}^m} \sum_{z\in\{0,1\}^{m_t}} \Pr[R = r] \Pr[U_1 = u]| \Pr[Z = z|_{R=r,U_1=u}] - \Pr[U_2 = z]| \end{aligned} \tag{3}$$

$$
\begin{aligned}
=&\frac{1}{2}\sum_{u\in\{0,1\}^m}\sum_{r\in\{0,1\}^r}\sum_{z\in\{0,1\}^{m_t}}\Pr[R=r]\Pr[U_1=u]|\Pr[Z=z|_{R=r,U_1=u}]-\Pr[U_2=z]|\\
=&\frac{1}{2}\sum_{u\in\{0,1\}^m}\Pr[U_1=u]\sum_{r\in\{0,1\}^r}\sum_{z\in\{0,1\}^{m_t}}\Pr[R=r]|\Pr[Z=z|_{R=r,U_1=u}]-\Pr[U_2=z]|\\
=&\frac{1}{2}\sum_{u\in\{0,1\}^m}\Pr[U_1=u]\sum_{r\in\{0,1\}^r}\sum_{z\in\{0,1\}^{m_t}}\Pr[R=r]|\Pr[\mathsf{Ext}_t(X_t,R)=z|_{R=r,Y=u}]-\Pr[U_2=z]|\\
=&\sum_{u\in\{0,1\}^m}\Pr[U_1=u]\mathsf{SD}(R\circ\mathsf{Ext}_t(X_t,R)|_{Y=u},R\circ U_2)\\
\leq&\sum_{u\in\{0,1\}^m}\Pr[U_1=u]\epsilon_t\\
=&\epsilon_t.
\end{aligned}
$$

So $\mathsf{SD}(R\circ Y\circ\mathsf{Ext}_t(X_t,R),R\circ U_1\circ U_2)\leq\sum_{i=1}^t\epsilon_t$. This proves the lemma. □

For any circuit C, the size of C is denoted as $\mathsf{size}(C)$. The depth of C is denoted as $\mathsf{depth}(C)$.

Definition 2.10 (AC^0)**.** AC^0 *is the complexity class which consists of all families of circuits having constant depth and polynomial size. The gates in those circuits are* NOT *gates,* AND *gates and* OR *gates where* AND *gates and* OR *gates have unbounded fan-in.*

Lemma 2.11. *The following are some well known properties of* AC^0 *circuits.*

1. *Any boolean function* $f:\{0,1\}^{l=\Theta(\log n)}\to\{0,1\}$ *can be computed by an* AC^0 *circuit of size* $\mathsf{poly}(n)$ *and depth* 2*. In fact, it can be represented by either a CNF or a DNF.*
2. *For every* $c\in\mathbb{N}$*, every integer* $l=\Theta(\log^c n)$*, the inner product function* $\langle\cdot,\cdot\rangle:\{0,1\}^l\times\{0,1\}^l\to\{0,1\}$ *can be computed by an* AC^0 *circuit of size* $\mathsf{poly}(n)$ *and depth* $c+1$*.*

Proof. For the first property, for an input string $u\in\{0,1\}^l$,

$$
f(u)=\bigvee_{j=0}^{2^l-1}(I_{u=\langle j\rangle}\wedge f(\langle j\rangle))=\bigwedge_{j=0}^{2^l-1}(I_{u\neq\langle j\rangle}\vee f(\langle j\rangle)).
$$

Here I_e is the indicator function such that $I_e=1$ if e is true and $I_e=0$ otherwise. We know that $I_{u=\langle j\rangle}$ can be represented as a boolean formula with only AND and NOT gates, checking whether $u=\langle j\rangle$ bit by bit. Similarly $I_{u\neq\langle j\rangle}$ can be represented as a boolean formula with only OR and NOT gates by taking the negation of $I_{u=\langle j\rangle}$. So the computation of obtaining $f(u)$ can be represented by a CNF/DNF. Thus it can be realized by a circuit of depth 2 by merging the gates of adjacent levels.

For the second property, assume we are computing $\langle s,x\rangle$. Consider a c-step algorithm which is as follows.

In the first step, we divide s into blocks $s_1,s_2,\ldots,s_t$, where $|s_i|=l'=\Theta(\log n),\forall i\in[t]$. Also we divide x into blocks $x_1,x_2,\ldots,x_t$, where $|x_i|=l',\forall i\in[t]$. Then we compute $\langle s_i,x_i\rangle$ for each $i\in[t]$ and provide them as the input bits for the next step. As each block has $\Theta(\log n)$ bits, this step can be done by a circuit of depth 2 according to the first property.

For the jth step where $j=2,\ldots,c$, we divide the input bits into blocks where each block has size l'. We compute the parity of the bits in each block and pass them to the next step as the inputs for the next step.

For the last step, if l' is large enough, there will be only 1 block and the parity of this block is $\langle s, x\rangle$.

For the jth step, where $j = 2, \ldots, c$, as the size of each block is $\Theta(\log n)$, we only need a circuit of depth 2 and polynomial size to compute the parity of each block according to the first property. The computation for all the blocks can be done in parallel. Thus the jth step can be computed by a circuit of depth 2 and polynomial size.

As a result, this algorithm can be realized by a circuit of depth $2c$ and polynomial size. By merging the gates of adjacent depths, we can have a circuit of depth $c + 1$ and polynomial size to compute $\langle s, x\rangle$.

□

Definition 2.12. *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 ($\mathsf{size}(C) \le g$), we have $\Pr_{x \leftarrow U}[C(x) = f(x)] < 1 - \delta$.*

Definition 2.13 (Graphs)**.** *Let $G = (V, E)$ be a graph. Let A be the adjacency matrix of G. Let $\lambda(G)$ be the second largest eigenvalue of A. We say G is d-regular, if the degree of G is d. When G is clear in the context, we simply denote $\lambda(G)$ as λ.*

3 Lower Bound for Error Parameters of AC^0 Extractors

Here we show a lower bound on the error of strong AC^0 seeded extractors. Our conclusion is mainly based on the well known LMN theorem deduced by Fourier analysis, given by Linial, Mansour, and Nisan [33].

Let the Fourier expansion of a function $f : \{-1,1\}^l \to \{-1,1\}$ be $f(x) = \sum_{S \subseteq [n]} \hat{f}_S \chi_S(x)$, where $\chi_S(x) = \prod_{i=1}^{l} x_i$. For any $f, g : \{-1,1\}^l \to \{-1,1\}$, $\langle f, g\rangle = \frac{1}{2^l} \sum_{x \in \{-1,1\}^l} f(x)g(x)$.

Theorem 3.1 (LMN Theorem [33])**.** *Let $f : \{-1,1\}^l \to \{-1,1\}$ be computable by AC^0 circuits of size $s > 1$ and depth d. Let $\epsilon \in (0, 1/2]$. There exists $t = O(\log(s/\epsilon))^{d-1} \cdot \log(1/\epsilon)$ s.t.*

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

Our first lower bound is as the follows.

Theorem 3.2. *Let $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^{l=\mathsf{poly}(n)} \to \{0,1\}^m$ be a strong $(k = n - 1, \epsilon)$-extractor which can be computed by AC^0 circuits of depth d. Then $\epsilon \ge 1/2^{O(\log^d n)}$.*

Proof. Without loss of generality, let $m = 1$. Let's transform the function space of Ext to $\{-1,1\}^{n+l} \to \{-1,1\}$, achieving function f. Let $\epsilon_0 = 1/2$. By Theorem 3.1, there exists $t = O(\log(s/\epsilon_0))^{d-1} \cdot \log(1/\epsilon_0) = O(\log^{d-1} n)$ s.t.

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

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

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

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

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 $\{-1,1\}^l$ 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. 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

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

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,

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

So

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

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

□

We also consider extractors for bit-fixing sources and give the following negative result on error.

Theorem 3.3. *Let* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^{l=\mathsf{poly}(n)} \to \{0,1\}^m$ *be a strong extractor with error* ϵ*, for any* $(n, k = n - \mathsf{poly}\log n)$*-bit-fixing sources, which can be computed by* AC^0 *circuits of depth* d*. Then* $\epsilon \geq 1/2^{O(\log^d n)}$.

The proof is slightly different from that of theorem 3.2.

Proof. Let $m = 1$ and also transform the function space of Ext to $\{-1, 1\}^{n+l} \to \{-1, 1\}$, achieving function f. Let $\epsilon_0 = 1/2$. By Theorem 3.1, there exists $t = O(\log(s/\epsilon_0))^{d-1} \cdot \log(1/\epsilon_0) = O(\log^{d-1} n)$ s.t.

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

Fix an $S = S_1 \cup S_2$, with $|S| \leq t$, where $S_1 \subseteq [n], S_2 \subseteq \{n+1, n+2, \ldots, n+l\}$. 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 $\{-1, 1\}^{n+l}$.

For $a \in \{-1, 1\}^{|S_1|}$, let X_a be a uniform distribution over $\{-1, 1\}^n$ conditioned on $X_{S_1} = a$. For $b \in \{-1, 1\}^{|S_2|}$, let R_b be a uniform distribution over $\{-1, 1\}^l$ conditioned on $R_{S_2} = b$. So $\chi_S(x \circ r) =$

$\prod_{i\in[|S_1|]} a_i \prod_{j\in[|S_2|]} b_j$ 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 extractor, R_b only blows up the error by at most $2^{|S|}$. X_a has entropy $n - |S_1| \geq n - t = n - \mathsf{poly}(\log n)$. So

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

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

So

$$\forall a \in \{-1,1\}^{|S_1|}, \forall b \in \{-1,1\}^{|S_2|}, |\Pr[f(X_a \circ R_b) \neq \prod_{i\in[|S_1|]} a_i \prod_{j\in[|S_2|]} b_j] - 1/2| \leq 2^{|S|}\epsilon.$$

Thus

$$\begin{aligned}
|\Pr_u[f(u) \neq \chi_S(u)] - 1/2| &= |\sum_{a\in\{-1,1\}^{|S_1|}} \sum_{b\in\{-1,1\}^{|S_2|}} \frac{1}{2^{|S|}}(\Pr[f(X_a \circ R_b) \neq \prod_{i\in[|S_1|]} a_i \prod_{j\in[|S_2|]} b_j] - 1/2)| \\
&\leq \sum_{a\in\{-1,1\}^{|S_1|}} \sum_{b\in\{-1,1\}^{|S_2|}} \frac{1}{2^{|S|}}|\Pr[f(X_a \circ R_b) \neq \prod_{i\in[|S_1|]} a_i \prod_{j\in[|S_2|]} b_j] - 1/2| \\
&\leq 2^{|S|}\epsilon.
\end{aligned} \tag{5}$$

Hence

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

As a result,

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

So

$$\epsilon \geq 2^{-(t+1)} \sqrt{\frac{1}{2\sum_{i=0}^{t}\binom{n}{i}}}.$$

As $\sum_{i=0}^{t}\binom{n}{i} \leq (\frac{en}{t})^t = 2^{O(t\log n)} = 2^{O(\log^d n)}$, $\epsilon \geq 2^{-O(\log^d n)}$.

□

4 The Basic Construction of Extractors in AC^0

Our basic construction is based on the general idea of I-W generator [25]. In [42], 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 [46] summarizes these results in details, and we review them again. The first step is established by Babai et al. [6], which is an amplification from worst-case hardness to mildly average-case hardness.

Lemma 4.1 ([6]). *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 [27].

Lemma 4.2 ([27]). *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 [25]. Their construction uses the following Nisan-Widgerson Generator [37] which is widely used in hardness amplification.

Definition 4.3 ((n,m,k,l)-design and Nisan-Widgerson Generator [37]). *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 [37] showed that the (n,m,k,l)-design can be constructed efficiently.

Lemma 4.4 (Implicit in [37]). *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.*

As we need the parameters to be concrete (while in [37] they use big-O notations), we prove it again.

Proof. Our algorithm will construct these S_is one by one. For S_1, we can choose an arbitrary subset of $[n]$ of size αl.

First of all, S_1 can constructed by choosing l elements from $[n]$.

Assume we have constructed $S_1, \ldots, S_{i-1}$, now we construct S_i. We first prove that S_i exists. Consider a random subset of size l from $[n]$. Let $H_{i,j} = |S_i \cap S_j|$. We know that $\mathbf{E}H_{i,j} = l^2/n$. As $n = \lfloor \frac{10l}{\alpha} \rfloor \in [\frac{10l}{\alpha} - 1, \frac{10l}{\alpha}]$, $\mathbf{E}H_{i,j} \in [\frac{\alpha l}{10}, \frac{\alpha l}{10} + 1]$.

So $\Pr[H_{i,j} \geq \alpha l] \leq \Pr[H_{i,j} \geq (1+9)(\mathbf{E}H_{i,j} - 1)]$

By the Chernoff bound,

$$\Pr[H_{i,j} \geq 10(\mathbf{E}H_{i,j} - 1)] \leq \Pr[H_{i,j} \geq 9\mathbf{E}H_{i,j}] \leq \exp\{-\frac{8\mathbf{E}H_{i,j}}{3}\} \leq \exp\{-\frac{4}{15}\alpha l\} \leq \exp\{-\frac{\alpha l}{4}\}$$

By the union bound,

$$\Pr[\forall j = 1, \ldots, i-1, H_{i,j} \leq \alpha l] \geq 1 - m \exp\{-\frac{\alpha l}{4}\} > 0.$$

This proves that there exists a proper S_i. As there are n bits totally, we can find it in time polynomial of 2^n. □

The following is the third step of hardness amplification.

Lemma 4.5 (Implicit in [25]). *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 $l' = |a| + |s| + |v_1| + |w| = \Theta(l)$.

The proof of Lemma 4.5 is in the Appendix.

The construction of the Impagliazzo Widgerson Generator [25] is as follows. Given the input $x \leftarrow X$, let $f : \{0,1\}^{\log n} \to \{0,1\}$ be such that $f(\langle a \rangle) = x_a, \forall a \in [n]$. Then we run the 3 amplification steps, Lemma 4.1, Lemma 4.2 (part1 for a constant number of times, part 2 for once) and Lemma 4.5 sequentially to get function f' from f. The generator $\mathsf{IW}(x, u) = \mathsf{NW}_{f',\mathcal{S}}(u)$. As pointed out by Trevisan [42], the function IW is a (k, ϵ)-extractor. Let's call it the IW-Extractor. It is implicit in [42] that the output length of the IW-Extractor is k^α and the statistical distance of $\mathsf{IW}(X, U)$ from uniform distributions is $\epsilon = 1/k^\beta$ for some $0 < \alpha, \beta < 1$. This can be verified by a detailed analysis of the IW-Extractor.

However, this construction is not in AC^0 because the first amplification step is not in AC^0.

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

Construction 4.6. *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 4.2 part 1 for c_2 times and run amplification step of Lemma 4.2 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 4.5 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 4.7. *In Construction 4.6, Ext_0 is a strong $(k, 2\epsilon)$ extractor.*

The proof follows from the "Bad Set" argument given by Trevisan [42]. In Trevisan [42] 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.

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

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 [37], we have the following lemma.

Lemma 4.8 (Implicit in [37] [42])**.** *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 4.8, 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 $1/2+\epsilon/m\ge 1/2+1/n^{1/400}$ fraction of inputs.

By Lemma 4.5, 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 4.2 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 $\sum_{i=0}^{n/(c_1\log^{c_2}n)}\binom{n}{i}\le(e\cdot c_1\log^{c_2}n)^{n/(c_1\log^{c_2}n)}=2^{O(n/\log^{c_2-1}n)}$ number of x. So

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

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

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

Then we know,

$$\begin{aligned}
&|\Pr[A(U_s\circ \mathsf{Ext}(X,U_s))=1]-\Pr[A(U)=1]|\\
=&\sum_{x\in B}\Pr[X=x]|\Pr[A(U_s\circ \mathsf{Ext}(x,U_s))=1]-\Pr[A(U)=1]|\\
&+\sum_{x\notin B}\Pr[X=x]|\Pr[A(U_s\circ \mathsf{Ext}(x,U_s))=1]-\Pr[A(U)=1]|\\
\le&2\epsilon.
\end{aligned}\tag{6}$$

□

Lemma 4.9. *The seed length of construction 4.6 is $\Theta(\log n)$.*

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 $(\lceil 10 l_3/c \rceil = \Theta(l_3), m, cl_3, l_3)$-design. So $d = \lfloor 10 l_3/c \rfloor = \Theta(l_3) = \Theta(\log n)$.

□

Lemma 4.10. *The function* Ext_0 *in Construction 4.6 is in* AC^0. *The circuit depth is* $c_2 + 5$. *The locality is* $\Theta(\log^{c_2+2} n) = \mathsf{poly}(\log n)$.

Proof. First we prove that the locality is $\Theta(\log^{c_2+2} n)$.

By the construction of f_1, we know $f_1(\langle i \rangle)$ is equal to the ith bit of x.

Fix the seed u. According to Lemma 4.2 part 1, if we apply the amplification once to get f' from f, then $f'(s, r)$ depends on $f(w_1), f(w_2), \cdots, f(w_l)$, as

$$f'(s, r) = \langle s, f(w_1) \circ f(w_2) \circ \cdots \circ f(w_l) \rangle.$$

Here $l = O(\log n)$ is equal to the input length of f.

The construction in Lemma 4.2 part 2 is the same as that of Lemma 4.2 part 1. As a result, if apply Lemma 4.2 part 1 for c_2 times and Lemma 4.2 part 2 for 1 time to get f_2 from f_1, the output of f_2 depends on $\Theta(\log^{c_2+1} n)$ bits of the input x.

According to Lemma 4.5, the output of f_3 depends on $f_2(a|_{S_1} \oplus v_1), f_2(a|_{S_2} \oplus v_2), \cdots, f_2(a|_{S_l} \oplus v_{l_2})$, as

$$f_3(a, s, v_1, w) = \langle s, f_2(a|_{S_1} \oplus v_1) \circ f_2(a|_{S_2} \oplus v_2) \circ \cdots f_2(a|_{S_l} \oplus v_{l_2}) \rangle$$

So the output of f_3 depends on $O(\log^{c_2+2} n)$ bits of the x.

So the overall locality is $O(\log^{c_2+2} n) = \mathsf{poly} \log n$.

Next we prove that the construction is in AC^0.

The input of Ext_0 has two parts, x and u. Combining all the hardness amplification steps and the NW generator, we can see that essentially u is used for two purposes: to select some $t = \Theta(\log^{c_2+2}(n))$ bits (denote it as x') from x (i.e., provide t indices $u'_1, \ldots, u'_t$ in $[n]$), and to provide a vector s' of length t, finally taking the inner product of x' and the vector s'. Here although for each amplification step we do an inner product operation, the overall procedure can be realized by doing only one inner product operation.

Since u has $O(\log n)$ bits, s' can be computed from u by using a circuit of depth 2, according to Lemma 2.11 part 1.

Next we show that selecting x' from x using the indices can be computed by CNF/DNFs, of polynomial size, with inputs being x and the indices. The indices, $u'_i, i \in [t]$, are decided by u. Let's assume $\forall i \in [t], u'_i = h_i(u)$ for some deterministic functions $h_i, i \in [t]$. As $|u| = O(\log n)$, the indices can be computed by CNF/DNFs of polynomial size. Also $\forall i \in [t], f(u'_i)$ can be represented by a CNF/DNF when u'_i is given. This is because

$$f(u'_i) = \bigvee_{j=0}^{|x|} (I_{u'_i=j} \wedge x_j) = \bigwedge_{j=0}^{|x|} (I_{u'_i \neq j} \vee x_j).$$

Here I_e is the indicator function such that $I_e = 1$ if e is true and $I_e = 0$ otherwise. We know that $I_{u'_i=j}$ can be represented by a boolean formula with only AND and NOT gates, checking whether $u'_i = j$ bit by bit. Similarly $I_{u'_i \neq j}$ can be represented by a boolean formula with only OR and NOT gates, taking the negation of $I_{u'_i=j}$. As a result, this step can be computed by a circuit of depth 2.

So the computation of obtaining x' can be realized by a circuit of depth 3 by merging the gates between adjacent depths.

Finally we can take the inner product of two vectors x' and s' of length $t = \Theta(\log^{c_2+2}(n))$. By Lemma 2.11 part 2, we know that this computation can be represented by a poly-size circuit of depth $c_2 + 3$.

The two parts of computation can be merged together to be a circuit of depth $c_2 + 5$, as we can merge the last depth of the circuit obtaining x' and the first depth of the circuit computing the inner product. The

size of the circuit is polynomial in n as both obtaining x' and the inner product operation can be realized by poly-size circuits.

□

According to to Lemma 4, Lemma 4.9, Lemma 4.10, we have the following theorem.

Theorem 4.11. *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 $\epsilon = n^{-1/600}$, $d = \Theta(\log n)$, $m = \lfloor n^{\frac{1}{3600}} \rfloor$ and the locality is $\Theta(\log^{c+4} n) = \mathsf{poly}\log n$.*

We call this extractor the Basic-AC^0-Extractor.

5 Error Reduction for AC^0 Extractors

5.1 For Polynomially Small Error

According to Theorem 4.11, for any $k = \frac{n}{\mathsf{poly}(\log n)}$, we have a (k, ϵ)-extractor in AC^0, with $\epsilon = 1/n^\beta$ where β is a constant. In this subsection, we will reduce the error parameter ϵ to give an explicit (k, ϵ)-extractor in AC^0 such that ϵ can be any $1/\mathsf{poly}(n)$.

The first tool we will use is the AC^0 extractor given by Goldreich et al. [19]. Although it's output length is only $O(\log n)$, the error parameter can be any $1/\mathsf{poly}(n)$.

Lemma 5.1 (Theorem 3.1 of Goldreich et al. [19])**.** *For every $k = \delta n = n/\mathsf{poly}(\log n)$ and every $\epsilon = 1/\mathsf{poly}(n)$, there exist an explicit construction of a strong extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^{O(\log n)} \to \{0,1\}^{\Theta(\log n)}$ which is in AC^0 of depth $4 + \lceil \frac{\log(n/k(n))}{\log\log n} \rceil$.*

If $\delta \in (0,1]$ is a constant, the locality of this extractor is $\Theta(\log n)$.

The construction of Theorem 5.1 is a classic sample-then-extract procedure following from Vadhan [44]. First they developed a sampling method in AC^0, then on input X they sample a source of length $\mathsf{poly}(\log n)$ with entropy rate $O(\delta)$. At last they use the extractor in [16] to finish the extraction.

Another tool we will be relying on is the error reduction method for extractors, given by Raz et al. [39]. They give an error reduction method for poly-time extractors and we will adapt it to the AC^0 settings.

Lemma 5.2 (G_x Property [39])**.** *Let $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be a (k, ϵ)-extractor with $\epsilon < 1/4$. Let X be any $(n, k+t)$-source. For every $x \in \{0,1\}^n$, there exists a set G_x such that the following holds.*

- *For every $x \in \{0,1\}^n$, $G_x \subset \{0,1\}^d$ and $|G_x|/2^d = 1 - 2\epsilon$.*
- *$\mathsf{Ext}(X, G_X)$ is within distance at most 2^{-t} from an $(m, m - O(1))$-source. Here $\mathsf{Ext}(X, G_X)$ is obtained by first sampling x according to X, then choosing r uniformly from G_x, and outputting $\mathsf{Ext}(x, r)$. We also denote $\mathsf{Ext}(X, G_X)$ as $\mathsf{Ext}(X, U)|_{U \in G_X}$.*

Raz et al. [39] showed the following result.

Lemma 5.3 ([39])**.** *Let $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be a (k, ϵ)-extractor. Consider $\mathsf{Ext}' : \{0,1\}^n \times \{0,1\}^{2d} \to \{0,1\}^{2m}$ which is constructed in the following way.*

$$\mathsf{Ext}'(x, u) = \mathsf{Ext}(x, u_1) \circ \mathsf{Ext}(x, u_2)$$

Here $u = u_1 \circ u_2$.

For any $t \le n - k$, let X be an $(n, k+t)$-source . Let U be the uniform distribution of length $2d$. With probability at least $1 - O(\epsilon^2)$, $\mathsf{Ext}'(X, U)$ is 2^{-t}-close to having entropy $m - O(1)$.

Remark 5.4. *Here we briefly explain the result in lemma 5.3. The distribution of* $Y = \mathsf{Ext}'(X, U_1 \circ U_2)$ *is the convex combination of* $Y|_{U_1 \in G_X, U_2 \in G_X}$, $Y|_{U_1 \notin G_X, U_2 \in G_X}$, $Y|_{U_1 \in G_X, U_2 \notin G_X}$ *and* $Y|_{U_1 \notin G_X, U_2 \notin G_X}$. *That is*

$$\begin{aligned} Y = & I_{U_1 \in G_X, U_2 \in G_X} Y|_{U_1 \in G_X, U_2 \in G_X} + I_{U_1 \notin G_X, U_2 \in G_X} Y|_{U_1 \notin G_X, U_2 \in G_X} \\ & + I_{U_1 \in G_X, U_2 \notin G_X} Y|_{U_1 \in G_X, U_2 \notin G_X} + I_{U_1 \notin G_X, U_2 \notin G_X} Y|_{U_1 \notin G_X, U_2 \notin G_X}. \end{aligned} \tag{7}$$

Also we know that $\Pr[I_{U_1 \notin G_X, U_2 \notin G_X} = 1] = O(\epsilon^2)$. *As a result, according to Lemma 5.2, this lemma follows.*

Informally speaking, this means that if view $Y = \mathsf{Ext}'(X, U) = Y_1 \circ Y_2$, then with high probability either Y_1 or Y_2 is 2^t-close to having entropy $m - O(1)$.

We adapt this lemma by doing the extraction for any $t \in \mathbb{N}^+$ times instead of 2 times. We have the following result.

Lemma 5.5. *Let* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *be a* (k, ϵ)*-extractor. For any* $t \in \mathbb{N}^+$, *consider* $\mathsf{Ext}' : \{0,1\}^n \times \{0,1\}^{td} \to \{0,1\}^{tm}$ *which is constructed in the following way.*

$$\mathsf{Ext}'(x, u) = \mathsf{Ext}(x, u_1) \circ \mathsf{Ext}(x, u_2) \circ \cdots \circ \mathsf{Ext}(x, u_t)$$

Here $u = u_1 \circ u_2 \circ \cdots \circ u_t$.

For any $a \leq n - k$, *let* X *be an* $(n, k + a)$*-source. Let* $U = \bigcirc_{i=1}^{t} U_i$ *be the uniform distribution such that* $\forall i \in [t], |U_i| = d$.

1. *For* $S \subseteq [t]$, *let* $I_{S,X}$ *be the indicator such that* $I_{S,X} = 1$ *if* $\forall i \in S, U_i \in G_X, \forall j \notin S, U_j \notin G_X$ *and* $I_{S,X} = 0$ *otherwise. Here* G_X *is defined according to 5.2. The distribution of* $\mathsf{Ext}'(X, U)$ *is a convex combination of the distributions of* $\mathsf{Ext}'(X, U)|_{I_{S,X}=1}, S \subseteq [t]$. *That is*

$$U \circ \mathsf{Ext}'(X, U) = \sum_{S \subseteq [t]} I_{S,X} U \circ \mathsf{Ext}'(X, U)|_{I_{S,X}=1}$$

2. *For every* $S \subseteq [t_1], S \neq \emptyset$, *there exists an* $i^* \in [t_1]$ *such that* $\mathsf{Ext}(X, U_{i^*})|_{I_{S,X}=1}$ *is* 2^{-a}*-close to having entropy* $m - O(1)$.

Proof. The first assertion is proved as the follows. By the definition of G_x of Lemma 5.2, for each fixed $x \in \mathsf{supp}(X)$, $\sum_{S \subseteq [t]} I_{S,x} = 1$ as for each i, $U_i \in G_x$ either happens or not. Also $I_{S,X}$ is a convex combination of $I_{S,x}, \forall x \in \mathsf{supp}(X)$. So $\sum_{S \subseteq [t]} I_{S,X} = \sum_{S \subseteq [t]} \sum_{x \in \mathsf{supp}(X)} I_{S,x} I_{X=x} = 1$. As a result, the assertion follows.

The second assertion is proved as the follows. For every $S \subseteq [t_1], S \neq \emptyset$, by the definition of $I_{S,X}$, there exists an $i^* \in [t_1]$, $U_{i^*} \in G_X$. By Lemma 5.2, $\mathsf{Ext}(X, U_{i^*})|_{U_{i^*} \in G_X} = \mathsf{Ext}(X, U_{i^*})|_{I_{S,X}=1}$ is 2^{-a}-close to having entropy $m - O(1)$.

□

Finally we consider the following construction of an error reduction procedure.

Construction 5.6 (Error Reduction)**.** *For any* $c, c_0 \in \mathbb{N}$, *let* k *be any* $\Theta(n/\log^c n)$ *and* ϵ *be any* $\Theta(1/n^{c_0})$. *Let* X *be an* (n, k)*-source. We construct a strong* (k, ϵ)*-extractor* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *where* $d = O(\log n), m = k^{\Theta(1)}$.

- *Let* $\mathsf{Ext}_0 : \{0,1\}^{n_0 = n} \times \{0,1\}^{d_0} \to \{0,1\}^{m_0}$ *be a* (k_0, ϵ_0)*-extractor following from Theorem 4.11 where* $k_0 = k - \Delta_1, \Delta_1 = \log(n/\epsilon)$, $\epsilon_0 = n^{-\Theta(1)}$, $d_0 = O(\log n)$, $m_0 = k^{\Theta(1)}$.

- *Let* $\mathsf{Ext}_1 : \{0,1\}^{n_1=m_0/t_2} \times \{0,1\}^{d_1} \to \{0,1\}^{m_1}$ *be a* (k_1, ϵ_1)*-extractor following from Lemma 5.1 where* $k_1 = 0.9n_1$*,* $\epsilon_1 = \epsilon/n$*,* $d_1 = O(\log n), m_1 = \Theta(\log n)$*.*
- *Let* t_1 *be such that* $(2\epsilon_0)^{t_1} \leq 0.1\epsilon$*. (We only consider the case that* $\epsilon < \epsilon_0$*. If* $\epsilon \geq \epsilon_0$*, we set* Ext *to be* Ext_0*.)*
- *Let* $t_2 = m_0^{1/3}$*.*

Our construction is as follows.

1. *Let* $R_1, R_2, \ldots, R_{t_1}$ *be independent uniform distributions such that for every* $i \in [t_1]$ *the length of* R_i *is* d_0*. Get* $Y_1 = \mathsf{Ext}_0(X, R_1), \ldots, Y_{t_1} = \mathsf{Ext}_0(X, R_{t_1})$*.*
2. *Get* $Y = Y_1 \circ Y_2 \circ Y_3 \circ \cdots \circ Y_{t_1}$*.*
3. *For each* $i \in [t_1]$*, let* $Y_i = Y_{i,1} \circ Y_{i,2} \circ \cdots \circ Y_{i,t_2}$ *such that for every* $j \in [t_2]$*,* $Y_{i,j}$ *has length* $n_1 = m_0/t_2$*. Let* $S_1, S_2, \ldots, S_{t_1}$ *be independent uniform distributions, each having length* d_1*. Get* $Z_{i,j} = \mathsf{Ext}_1(Y_{i,j}, S_i), \forall i \in [t_1], j \in [t_2]$*. Let* $Z_i = Z_{i,1} \circ Z_{i,2} \circ \cdots Z_{i,t_2}$*.*
4. *Let* $R = \bigcirc_i R_i, S = \bigcirc_i S_i$*. We get* $\mathsf{Ext}(X, U) = Z = \bigoplus_i^{t_1} Z_i$ *where* $U = R \circ S$*.*

Lemma 5.7. *Construction 5.6 gives a strong* (k, ϵ)*-extractor.*

Lemma 5.8 (Chain Rule of Min-Entropy [45])**.** *Let* (X, Y) *be a jointly distributed random variable with entropy* k*. The length of* X *is* l*. For every* $\epsilon > 0$*, with probability at least* $1 - \epsilon$ *over* $x \leftarrow X$*,* $Y|_{X=x}$ *has entropy* $k - l - \log(1/\epsilon)$*.*

Also there exists another source (X, Y') *such that* $\forall x \in \{0,1\}^l$*,* $Y'|_{X=x}$ *has entropy* $k - l - \log(1/\epsilon)$ *and* $\mathsf{SD}((X, Y), (X, Y')) \leq \epsilon$*.*

Lemma 5.9. *Let* $X = X_1 \circ \cdots \circ X_t$ *be an* $(n, n - \Delta)$*-source where for each* $i \in [t]$*,* $|X_i| = n_1 = \omega(\Delta)$*.*
Let $k_1 = n_1 - \Delta - \log(1/\epsilon_0)$ *where* ϵ_0 *can be as small as* $1/2^{0.9n_1}$*.*
Let $\mathsf{Ext}_1 : \{0,1\}^{n_1} \times \{0,1\}^{d_1} \to \{0,1\}^{m_1}$ *be a strong* (k_1, ϵ_1)*-extractor.*
Let $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *be constructed as the following,*

$$\mathsf{Ext}(X, U_s) = \mathsf{Ext}_1(X_1, U_s) \circ \cdots \circ \mathsf{Ext}_1(X_t, U_s).$$

Then Ext *is a strong* $(n - \Delta, \epsilon)$*-extractor where* $\epsilon = \mathsf{SD}(U_s \circ \mathsf{Ext}(X, U_s), U) \leq t(\epsilon_0 + \epsilon_1)$*.*

Proof. We prove by induction over the block index i.

For simplicity, let $\tilde{X}_i = X_1 \circ \cdots \circ X_i$ for every i. We slightly abuse the notation Ext here so that $\mathsf{Ext}(\tilde{X}_i, U_s) = \mathsf{Ext}_1(X_1, U_s) \circ \cdots \circ \mathsf{Ext}_1(X_i, U_s)$ denotes the extraction for the first i blocks.

For the first block, we know $H_\infty(X_1) = n_1 - \Delta$. According to the definition of Ext_1,

$$\mathsf{SD}(U_s \circ \mathsf{Ext}_1(X_1, U_s), U) \leq \epsilon_1 \leq (\epsilon_0 + \epsilon_1).$$

Assume for the first $i - 1$ blocks, $\mathsf{SD}(U_s \circ \mathsf{Ext}_1(\tilde{X}_{i-1}, U_s), U) \leq (i-1)(\epsilon_0 + \epsilon_1)$. Consider $\tilde{X}_i$. By Lemma 5.8, we know that there exists X_i' such that $\mathsf{SD}(\tilde{X}_i, \tilde{X}_{i-1} \circ X_i') \leq \epsilon_0$, where X_i' is such that $\forall \tilde{x}_{i-1} \in \mathsf{supp}(\tilde{X}_{i-1}), H_\infty(X_i'|\tilde{X}_{i-1} = \tilde{x}_{i-1}) \geq n_1 - \Delta - \log(1/\epsilon_0)$. So according to Lemma 2.4 part 2, as $U_s \circ \mathsf{Ext}(\tilde{X}_{i-1}, U_s) \circ \mathsf{Ext}_1(X_i, U_s)$ is a convex combination of $u \circ \mathsf{Ext}(\tilde{X}_{i-1}, u) \circ \mathsf{Ext}_1(X_i, u), \forall u \in \mathsf{supp}(U_s)$ and $U_s \circ \mathsf{Ext}(\tilde{X}_{i-1}, U_s) \circ \mathsf{Ext}_1(X_i', U_s)$ is a convex combination of $u \circ \mathsf{Ext}(\tilde{X}_{i-1}, u) \circ \mathsf{Ext}_1(X_i', u), \forall u \in \mathsf{supp}(U_s)$, we have

$$\mathsf{SD}(U_s \circ \mathsf{Ext}(\tilde{X}_{i-1}, U_s) \circ \mathsf{Ext}_1(X_i, U_s), U_s \circ \mathsf{Ext}(\tilde{X}_{i-1}, U_s) \circ \mathsf{Ext}_1(X_i', U_s)) \leq \mathsf{SD}(\tilde{X}_i, \tilde{X}_{i-1} \circ X_i') \leq \epsilon_0.$$

According to the assumption, Lemma 2.9 and the triangle inequality of Lemma 2.4, we have the following.

$$\begin{aligned}
&\mathsf{SD}(U_s \circ \mathsf{Ext}(\tilde{X}_{i-1}, U_s) \circ \mathsf{Ext}_1(X_i, U_s), U) \\
\leq &\mathsf{SD}(U_s \circ \mathsf{Ext}(\tilde{X}_{i-1}, U_s) \circ \mathsf{Ext}_1(X_i, U_s), U_s \circ \mathsf{Ext}(\tilde{X}_{i-1}, U_s) \circ \mathsf{Ext}_1(X'_i, U_s)) \\
&+ \mathsf{SD}(U_s \circ \mathsf{Ext}(\tilde{X}_{i-1}, U_s) \circ \mathsf{Ext}_1(X'_i, U_s), U) \\
\leq &\epsilon_0 + (i-1)(\epsilon_0 + \epsilon_1) + \epsilon_1 \\
= &i(\epsilon_0 + \epsilon_1)
\end{aligned} \tag{8}$$

The first inequality is due to the triangle property of Lemma 2.4. For the second inequality, first we have already shown that $\mathsf{SD}(U_s \circ \mathsf{Ext}(\tilde{X}_{i-1}, U_s) \circ \mathsf{Ext}_1(X_i, U_s), U_s \circ \mathsf{Ext}(\tilde{X}_{i-1}, U_s) \circ \mathsf{Ext}_1(X'_i, U_s)) \leq \epsilon_0$. Second, as $\tilde{X}_{i-1} \circ X'_i$ is a $((i-1)n_1 - \Delta, n_1 - \Delta - \log(1/\epsilon_0))$-block source, by our assumption and Lemma 2.9, $\mathsf{SD}(U_s \circ \mathsf{Ext}(\tilde{X}_{i-1}, U_s) \circ \mathsf{Ext}_1(X'_i, U_s), U) \leq (i-1)(\epsilon_0 + \epsilon_1) + \epsilon_1$. This proves the induction step.

As a result, $\mathsf{SD}(U_s \circ \mathsf{Ext}(X, U_s), U) \leq (\epsilon_0 + \epsilon_1)t$. □

Lemma 5.10. *In Construction 5.6, the output length of* Ext *is* $m = \Theta(m_0^{1/3} \log n) = \Theta(n^{10800} \log n)$.

Proof. The output length is equal to $t_2 \times m_1 = m_0^{1/3}\Theta(\log n) = \Theta(n^{1/10800} \log n)$

□

Next we prove Lemma 5.7.

Proof of Lemma 5.7. By Lemma 5.5,

$$\begin{aligned}
&R \circ Y \\
= &\sum_{T \subseteq [t_1]} I_{T,X}(R \circ Y|_{I_{T,X}=1}) \\
= &I_{\emptyset,X}(R \circ Y|_{I_{\emptyset,X}=1}) + (1 - I_{\emptyset,X})(R \circ Y|_{I_{\emptyset,X}=0}) \\
= &I_{\emptyset,X}(R \circ Y|_{I_{\emptyset,X}=1}) + \sum_{T \subseteq [t_1], T \neq \emptyset} I_{T,X}(R \circ Y|_{I_{T,X}=1})
\end{aligned} \tag{9}$$

where $I_{T,X}$ is the indicator such that $I_{T,X} = 1$, if $\forall i \in T, R_i \in G_X, \forall i \notin T, R_i \notin G_X$ and $I_{T,X} = 0$, otherwise. The G_X here is defined by Lemma 5.2 on X and Ext_0.

Fixing a set $T \subseteq [t_1], T \neq \emptyset$, by Lemma 5.5, there exists an $i^* \in [t_1]$ such that $R_{i^*} \in G_X$ and $R \circ Y|_{I_{T,X}=1}$ is $2^{-\Delta_1}$-close to

$$R' \circ A \circ W \circ B = \bigcirc_i R'_i \circ A \circ W \circ B$$

where $W = Y_{i^*}|_{R_i \in G_X}$ has entropy at least $m_0 - O(1)$. Here A, B and $R'_i, i = 1, 2, \ldots, t$ are some random variables where $A = (i^* - 1)m_1, |B| = (t - i^*)m_1$ and $\forall i \in [t], |R'_i| = d_1$. In fact, $R' = R|_{I_{T,X}=1}$.

According to our construction, next step we view $A \circ W \circ B$ as having $t_1 t_2$ blocks of block size n_1. We apply the extractor Ext_1 on each block. Although for all blocks the extractions are conducted simultaneously, we can still view the procedure as first extracting A and B, then extracting W. Assume for A, after extraction by using seed S_A, it outputs A'. Also for B, after extraction by using seed S_B, it outputs B'. So after extracting A and B, we get $A' \circ W \circ B'$. The length of $A' \circ B'$ is at most $t_1 m_0 m_1 / n_1 = t_1 t_2 m_1 = \Theta(t_1 t_2 \log n)$, as $n_1 = m_0 / t_2$.

We know that $n_1 = m_0/t_2 = m_0^{2/3} \geq 10 t_1 t_2 m_1 = m_0^{1/3}\mathsf{poly}(\log n)$. Also according to Lemma 5.8, $R' \circ S_A \circ S_B \circ A' \circ W \circ B'$ is ϵ'-close to $R' \circ S_A \circ S_B \circ A' \circ W' \circ B'$ such that for every $r' \in$

$\mathsf{supp}(R'), a \in \mathsf{supp}(A'), b \in \mathsf{supp}(B'), s_A \in \mathsf{supp}(S_A), s_B \in \mathsf{supp}(S_B)$, conditioned on $R' = r', S_A = s_A, S_B = s_B, A' = a, B' = b$, W' has entropy at least $n_1 - O(\log n) - t_1 t_2 m_1 - \log(1/\epsilon') = n_1 - \Delta_2$ where $\Delta_2 = O(\log n) + t_1 t_2 m_1 + \log(1/\epsilon') = O(m_0^{1/3} \log n)$. Here ϵ' can be as small as $2^{-k^{\Omega(1)}}$. That is

$$\begin{aligned}&\forall r' \in \mathsf{supp}(R'), a \in \mathsf{supp}(A'), b \in \mathsf{supp}(B'), s_A \in \mathsf{supp}(S_A), s_B \in \mathsf{supp}(S_B),\\ &H_\infty(W'|_{R'=r', S_A=s_A, S_B=s_B, A'=a, B'=b}) \geq n_1 - \Delta_2.\end{aligned} \tag{10}$$

Let $\mathsf{Ext}_1'(W', S_{i^*}) = \bigcirc_{i\in[t_2]}\mathsf{Ext}_1(W_i', S_{i^*})$ where $W' = \bigcirc_{i\in[t_2]} W_i'$ and $\forall i \in [t_2], |W_i'| = n_1$. By Lemma 5.9, as $k_1 = 0.9n_1 \leq n_1 - \Delta_2$, $S_{i^*} \circ \mathsf{Ext}_1'(W', S_{i^*})|_{R'=r', S_A=s_A, S_B=s_B, A'=a, B'=b}$ is $(\epsilon_0' + \epsilon_1)t_2$-close to uniform distributions where ϵ_0' can be as small as $2^{-k^{\Omega(1)}}$.

As a result, we have the following.

$$\begin{aligned}&\mathsf{SD}(U \circ \mathsf{Ext}(X, U), U')\\ =&\mathsf{SD}(R \circ S \circ \mathsf{Ext}(X, U), R \circ S \circ \tilde{U})\\ =&\mathsf{SD}(I_{\emptyset,X}(R \circ S \circ \mathsf{Ext}(X, U)|_{I_{\emptyset,X}=1}), I_{\emptyset,X}(R \circ S \circ \tilde{U}|_{I_{\emptyset,X}=1}))\\ &+ \mathsf{SD}((1 - I_{\emptyset,X})(R \circ S \circ \mathsf{Ext}(X, U)|_{I_{\emptyset,X}=0}), (1 - I_{\emptyset,X})(R \circ S \circ \tilde{U}|_{I_{\emptyset,X}=0}))\\ =&\Pr[I_{\emptyset,X} = 1]\mathsf{SD}(R \circ S \circ \mathsf{Ext}(X, U)|_{I_{\emptyset,X}=1}, R \circ S \circ \tilde{U}|_{I_{\emptyset,X}=1})\\ =&(2\epsilon_0)^{t_1}\mathsf{SD}(R \circ S \circ \mathsf{Ext}(X, U)|_{I_{\emptyset,X}=1}, R \circ S \circ \tilde{U}|_{I_{\emptyset,X}=1})\\ &+ \mathsf{SD}((1 - I_{\emptyset,X})(R \circ S \circ \mathsf{Ext}(X, U)|_{I_{\emptyset,X}=0}), (1 - I_{\emptyset,X})(R \circ S \circ \tilde{U}|_{I_{\emptyset,X}=0}))\end{aligned} \tag{11}$$

As

$$(2\epsilon_0)^{t_1}\mathsf{SD}(R \circ S \circ \mathsf{Ext}(X, U)|_{I_{\emptyset,X}=1}, R \circ S \circ \tilde{U}|_{I_{\emptyset,X}=1}) \leq (2\epsilon_0)^{t_1}$$

let's focus on $\mathsf{SD}((1 - I_{\emptyset,X})(R \circ S \circ \mathsf{Ext}(X, U)|_{I_{\emptyset,X}=0}), (1 - I_{\emptyset,X})(R \circ S \circ \tilde{U}|_{I_{\emptyset,X}=0}))$.

$$\begin{aligned}&\mathsf{SD}((1 - I_{\emptyset,X})(R \circ S \circ \mathsf{Ext}(X, U)|_{I_{\emptyset,X}=0}), (1 - I_{\emptyset,X})(R \circ S \circ \tilde{U}|_{I_{\emptyset,X}=0}))\\ =&\mathsf{SD}(\sum_{T\subseteq[t_1],T\neq\emptyset} I_{T,X}(R \circ S \circ \mathsf{Ext}(X, U)|_{I_{T,X}=1}), \sum_{T\subseteq[t_1],T\neq\emptyset} I_{T,X}(R \circ S \circ \tilde{U}|_{I_{T,X}=1}))\\ =&\sum_{T\subseteq[t_1],T\neq\emptyset} \Pr[I_{T,X} = 1]\mathsf{SD}(R \circ S \circ \mathsf{Ext}(X, U)|_{I_{T,X}=1}, R \circ S \circ \tilde{U}|_{I_{T,X}=1})\\ \leq&\sum_{T\subseteq[t_1],T\neq\emptyset} \Pr[I_{T,X} = 1](2^{-\Delta_1} + \mathsf{SD}(R' \circ S \circ (A' \oplus \mathsf{Ext}_1'(W, S_{i^*}) \oplus B'), R' \circ S \circ \tilde{U}))\\ =&(1 - (2\epsilon_0)^{t_1})(2^{-\Delta_1} + \mathsf{SD}(R' \circ S \circ (A' \oplus \mathsf{Ext}_1'(W, S_{i^*}) \oplus B'), R' \circ S \circ \tilde{U}))\\ \leq&2^{-\Delta_1} + \mathsf{SD}(R' \circ S \circ (A' \oplus \mathsf{Ext}_1'(W, S_{i^*}) \oplus B'), R' \circ S \circ \tilde{U})\\ \leq&2^{-\Delta_1} + \mathsf{SD}(R' \circ S \circ (A' \oplus \mathsf{Ext}_1'(W, S_{i^*}) \oplus B'), R' \circ S \circ (A' \oplus \mathsf{Ext}_1'(W', S_{i^*}) \oplus B'))\\ &+ \mathsf{SD}(R' \circ S \circ (A' \oplus \mathsf{Ext}_1'(W', S_{i^*}) \oplus B'), R' \circ S \circ \tilde{U})\\ \leq&2^{-\Delta_1} + \epsilon' + \mathsf{SD}(R' \circ S \circ (A' \oplus \mathsf{Ext}_1'(W', S_{i^*}) \oplus B'), R' \circ S \circ \tilde{U})\\ \leq&2^{-\Delta_1} + \epsilon' + (\epsilon_0' + \epsilon_1)t_2\end{aligned} \tag{12}$$

Here $U, U', \tilde{U}$ are uniform distributions. In the second equation, $I_{\emptyset,X}$ is the indicator such that $I_{\emptyset,X} = 1$ if $\forall i \in [t_1], R_i \notin G_X$ where G_X is defined by Lemma 5.2 on X and Ext_0. For the first inequality, we need to show that

$$\mathsf{SD}(R \circ S \circ \mathsf{Ext}(X, U)|_{I_{T,X}=1}, R' \circ S \circ (A' \oplus \mathsf{Ext}_1'(W, S_{i^*}) \oplus B')) \leq 2^{-\Delta_1}.$$

We know that for every $s \in \mathsf{supp}(S)$, by Lemma 2.4 part 2,

$$\begin{aligned}&\mathsf{SD}(R \circ S \circ \mathsf{Ext}(X,U)|_{I_{T,X}=1,S=s}, R' \circ S \circ (A' \oplus \mathsf{Ext}'_1(W, S_{i^*}) \oplus B')|_{S=s})\\ \leq &\mathsf{SD}(R \circ Y|_{I_{T,X}=1}, R' \circ A \circ W \circ B)\\ \leq &2^{-\Delta_1}.\end{aligned} \tag{13}$$

Here $R \circ S \circ \mathsf{Ext}(X,U)|_{I_{T,X}=1,S=s} = h(R \circ Y|_{I_{T,X}=1})$ for some deterministic function h as $S = s$ is fixed. Also $R' \circ S \circ (A' \oplus \mathsf{Ext}'_1(W, S_{i^*}) \oplus B')|_{S=s} = h(R' \circ A \circ W \circ B)$ for the same reason. As a result,

$$\begin{aligned}&\mathsf{SD}(R \circ S \circ \mathsf{Ext}(X,U)|_{I_{T,X}=1}, R' \circ S \circ (A' \oplus \mathsf{Ext}'_1(W, S_{i^*}) \oplus B'))\\ = &\sum_{s \in \mathsf{supp}(S)} \Pr[S=s]\mathsf{SD}(R \circ S \circ \mathsf{Ext}(X,U)|_{I_{T,X}=1,S=s}, R' \circ S \circ (A' \oplus \mathsf{Ext}'_1(W, S_{i^*}) \oplus B')|_{S=s})\\ \leq &2^{-\Delta_1}.\end{aligned} \tag{14}$$

The third inequality holds by the triangle property of Lemma 2.4 part 1. The 4th inequality holds because by Lemma 2.4 part 2,

$$\begin{aligned}&\mathsf{SD}(R' \circ S \circ (A' \oplus \mathsf{Ext}'_1(W, S_{i^*}) \oplus B')|_{S=s}, R' \circ S \circ (A' \oplus \mathsf{Ext}'_1(W', S_{i^*}) \oplus B')|_{S=s})\\ \leq &\mathsf{SD}(R' \circ S \circ A \circ W \circ B, R' \circ S \circ A \circ W' \circ B)\\ \leq &\epsilon'.\end{aligned} \tag{15}$$

.

As a result, the total error is at most

$$(2\epsilon_0)^{t_1} + (2^{-\Delta_1} + \epsilon' + (\epsilon'_0 + \epsilon_1)t_2)$$

We can set $\epsilon' = 0.1\epsilon, \epsilon'_0 = \epsilon/n$ so that $(2^{-\Delta_1} + \epsilon' + (\epsilon'_0 + \epsilon_1)t_2) \leq 0.1\epsilon$. As $(2\epsilon_0)^{t_1} < 0.1\epsilon$, we know $\mathsf{SD}(U \circ \mathsf{Ext}(X,U), U') \leq \epsilon$.

□

Lemma 5.11. *In Construction 5.6, the function* Ext *can be realized by a circuit of depth* $c + 10$*. Its locality is* $\Theta(\log^{c+5} n) = \mathsf{poly}(\log n)$.

Proof. According to Theorem 4.11 and Lemma 5.1, both Ext_0 and Ext_1 in our construction are in AC^0. For Ext_0, it can be realized by circuits of depth $c + 5$. For Ext_1, it can be realized by circuits of depth $4 + \lceil \frac{\log(n_1/k_1)}{\log\log n_1} \rceil$. As $k_1 = \delta_1 n_1$ where δ_1 is a constant, the depth is in fact 5.

In the first and second steps of Construction 5.6, we only run Ext_0 for t_1 times in parallel. So the computation can be realized by circuits of depth $c + 7$

For the third step, we run Ext_1 for $t_1 t_2$ times in parallel, which can be realized by circuits of depth 5.

The last step, according to Lemma 2.11, taking the XOR of a constant number of bits can be realized by circuits of depth 2. Each bit of Z is the XOR of t_1 bits and all the bits of Z can be computed in parallel. So the computations in this step can be realized by circuits of depth 2.

Now we merge the three parts of circuits together. As the circuits between each parts can be merged by deleting one depth, our construction can be realized by circuits of depth

$$(c + 5) + 5 + 2 - 2 = c + 10.$$

For the locality, according to Theorem 4.11, the locality of Ext_0 is $\mathsf{poly}(\log n)$. According to Lemma 5.1, the locality of Ext_1 is $\Theta(\log n)$. So each bit of Z is related with at most $t_1 \times \Theta(\log n) \times \Theta(\log^{c+4} n) = \Theta(\log^{c+5} n) = \mathsf{poly}(\log n)$ bits of X. So the locality is $\Theta(\log^{c+5} n) = \mathsf{poly}(\log n)$.

□

Lemma 5.12. *In Construction 5.6,* $d = O(t_1(d_0 + d_1)) = O(\log n)$.

Proof. In Construction 5.6, as

$$U = R \circ S = \bigcirc_i R_i \circ \bigcirc_i S_i,$$

$|U| = O(t_1 d_0 + t_1 d_1)$. According to the settings of Ext_0 and Ext_1, we know that $d_0 = O(\log n)$ and $d_1 = O(\log n)$. Also we know that $t_1 = O(1)$ as $\epsilon_0 = n^{-\Theta(1)}$ and $\epsilon = 1/\Theta(n^{c_0})$. So $d = O(\log n)$. □

Theorem 5.13. *For any constant* $c \in \mathbb{N}$*, any* $k = \Theta(n/\log^c n)$ *and any* $\epsilon = 1/\mathsf{poly}(n)$*, there exists an explicit construction of a strong* (k, ϵ)*-extractor* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *in* AC^0 *of depth* $c + 10$ *where* $d = O(\log n)$ *,* $m = \Theta(n^{1/10800} \log n) = k^{\Theta(1)}$ *and the locality is* $\Theta(\log^{c+5} n) = \mathsf{poly}(\log n)$.

Proof. It follows from Construction 5.6, Lemma 5.7, Lemma 5.10 , Lemma 5.11 and Lemma 5.12.

□

5.2 For Super-Polynomially Small Error

By using poly-log length seeds, we can achieve even smaller errors. We mainly improve the sample-then-extract procedure.

We first analyze the sampling method which is well studied by Zuckerman [49], Vadhan [44], Goldreich et al. [19], etc.

Definition 5.14 ([44])**.** *A* (μ_1, μ_2, γ)*-averaging sampler is a function* $\mathsf{Samp} : \{0,1\}^r \to [n]^t$ *such that* $\forall f : [n] \to [0,1]$*, if* $\mathbf{E}_{i \in [n]}[f(i)] \geq \mu_1$*, then*

$$\Pr_{I \leftarrow \mathsf{Samp}(U_r)}[\frac{1}{t}\sum_{i \in I} f(i) < \mu_2] \leq \gamma.$$

The t *samples generated by the sampler must be distinct.*

According to Vadhan [44], we have the following lemma.

Lemma 5.15 (Sample a Source [44])**.** *Let* $0 < 3\tau \leq \delta \leq 1$*. If* $\mathsf{Samp} : \{0,1\}^r \to [n]^t$ *is a* (μ_1, μ_2, γ)*-averaging sampler for* $\mu_1 = (\delta - 2\tau)/\log(1/\tau)$ *and* $\mu_2 = (\delta - 3\tau)/\log(1/\tau)$*, then for every* $(n, \delta n)$*-source* X*, we have* $\mathsf{SD}(U \circ X_{\mathsf{Samp}(U_r)}, U \circ W) \leq \gamma + 2^{-\Omega(\tau n)}$*. Here* U *is the uniform distribution over* $\{0,1\}^r$*. For every* a *in* $\{0,1\}^r$*, the random variable* $W|_{U=a}$ *is a* $(t, (\delta - 3\tau)t)$*-source.*

In fact, Zuckerman [49] has already given a very good sampler (oblivious sampler) construction. This construction is based on the existence of randomness extractors.

Definition 5.16 ([49])**.** *An* $(n, m, t, \gamma, \epsilon)$*-oblivious sampler is a deterministic function* $\mathsf{Samp} : \{0,1\}^n \to (\{0,1\}^m)^t$ *such that* $\forall f : \{0,1\}^m \to [0,1]$*,*

$$\Pr_{I \leftarrow \mathsf{Samp}(U_r)}[|\frac{1}{t}\sum_{i \in I} f(i) - \mathbf{E}f| > \epsilon] \leq \gamma.$$

The following lemma explicitly gives a construction of oblivious samplers using extractors.

Lemma 5.17 ([49])**.** *If there is an explicit* $(k = \delta n, \epsilon)$*-extractor* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$*, then there is an explicit* $(n, m, t = 2^d, \gamma = 2^{1-(1-\delta)n}, \epsilon)$*-oblivious sampler.*

The sampler is constructed as follows. Given a seed x *of length* n*, the* $t = 2^d$ *samples are* $\mathsf{Ext}(x, u)$*,* $\forall u \in \{0,1\}^d$*.*

As a result, we can construct the following samplers.

Lemma 5.18. *For any $a \in \mathbb{N}^+$, let γ be any $1/2^{\Theta(\log^a n)}$.*

- *For any $c \in \mathbb{N}$, let ϵ be any $\Theta(1/\log^c n)$. There exists an explicit $(\Theta(\log^a n), \log n, t, \gamma, \epsilon)$-oblivious sampler for any integer $t \in [t_0, n]$ with $t_0 = \mathsf{poly}(\log n)$.*
- *For any constant α in $(0, 1)$, any $c \in \mathbb{N}$, any $\mu = \Theta(1/\log^c n)$, there exists an explicit $(\mu, \alpha\mu, \gamma)$-averaging sampler $\mathsf{Samp} : \{0,1\}^{\Theta(\log^a n)} \to [n]^t$ in AC^0 of circuit depth $a + 2$, for any integer $t \in [t_0, n]$ with $t_0 = \mathsf{poly}(\log n)$.*

  Specifically, if $c = 0$, t can be any integer in $[t_0, n]$ with $t_0 = (\log n)^{\Theta(a)}$.

Proof. Let $k = \log^a n$. For any $\epsilon = \Theta(1/\log^c n)$, let's consider a (k, ϵ)-extractor $\mathsf{Ext} : \{0,1\}^{n'=c_0 \log^a n} \times \{0,1\}^d \to \{0,1\}^{\log n}$ for some constant c_0, following Lemma 2.8. Here we make one modification. We replace the last d bits of the output with the seed. We can see in this way, Ext is still an extractor.

Here the entropy rate is $\delta = 1/c_0$ which is a constant. According to Lemma 2.8, we know that, d can be $\Theta(\frac{\log^2(n'/\epsilon)}{\log n'}) = \Theta(\log\log n)$.

For the first assertion, according to Lemma 5.17, there exists an explicit construction of a $(c_0 \log^a n, \log n, t, \gamma, \epsilon)$-oblivious sampler where $\gamma = 2^{1-(1-2/c_0)(c_0 \log^a n)}$. As we can increase the seed length to $\log n$ by padding uniform random bits, t can be any integer in $[t_0, n]$ with $t_0 = 2^{\Theta(\frac{\log^2(n'/\epsilon)}{\log n'})} = \mathsf{poly}(\log n)$. As c_0 can be any large enough constant, γ can be $1/2^{\Theta(\log^a n)}$.

Next we prove the second assertion.

According to the definition of oblivious sampler, we know that $\forall f : [n] \to [0, 1]$,

$$\Pr_{I \leftarrow \mathsf{Samp}(U)}[|\frac{1}{t}\sum_{i \in I} f(i) - \mathbf{E}f| > \epsilon] \le \gamma.$$

Next we consider the definition of averaging sampler.

Let $(1 - \alpha)\mu = \epsilon$. As $\mu = \Theta(1/\log^c n)$, $\epsilon = \Theta(1/\log^c n)$. For any $f : [n] \to [0, 1]$ such that $\mu \le \mathbf{E}f$, we have the following inequalities, where Samp is a $(c \log^a n, \log n, t, \gamma, \epsilon)$-oblivious sampler.

$$\begin{aligned}
&\Pr_{I \leftarrow \mathsf{Samp}(U)}[\frac{1}{t}\sum_{i \in I} f(i) < \alpha\mu] \\
=&\Pr_{I \leftarrow \mathsf{Samp}(U)}[\frac{1}{t}\sum_{i \in I} f(i) < \mu - \epsilon] \\
=&\Pr_{I \leftarrow \mathsf{Samp}(U)}[\mu - \frac{1}{t}\sum_{i \in I} f(i) > \epsilon] \\
\le&\Pr_{I \leftarrow \mathsf{Samp}(U)}[\mathbf{E}f - \frac{1}{t}\sum_{i \in I} f(i) > \epsilon] \\
\le&\Pr_{I \leftarrow \mathsf{Samp}(U)}[|\frac{1}{t}\sum_{i \in I} f(i) - \mathbf{E}f| > \epsilon] \\
\le&\gamma
\end{aligned} \tag{16}$$

The first inequality holds because if the event that $\mu - \frac{1}{t}\sum_{i \in I} f(i) > \epsilon$ happens, then the event that $\mathbf{E}f - \frac{1}{t}\sum_{i \in I} f(i) > \epsilon$ will happen, as $\mu \le \mathbf{E}f$. The second inequality is because $\mathbf{E}f - \frac{1}{t}\sum_{i \in I} f(i) \le |\mathbf{E}f - \frac{1}{t}\sum_{i \in I} f(i)|$. So if $\mathbf{E}f - \frac{1}{t}\sum_{i \in I} f(i) > \epsilon$ happens, then $|\frac{1}{t}\sum_{i \in I} f(i) - \mathbf{E}f| > \epsilon$ happens.

Also as we replace the last d bits of the output of our extractor with the seed, the samples are distinct according to the construction of Lemma 5.17.

According to the definition of averaging sampler, we know that this gives an explicit $(\mu, \alpha\mu, \gamma)$-averaging sampler.

According to the construction described in the proof of Lemma 5.17, the output of the sampler is computed by running the extractor following Lemma 2.8 for t times in parallel. So the circuit depth is equal to the circuit depth of the extractor Ext.

Let's recall the construction of the Trevisan's extractor Ext.

The encoding procedure is doing the multiplication of the encoding matrix and the input x of length $n' = c\log^a n$. By Lemma 2.11, this can be done by a circuit of depth $a+1$.

The last step is the procedure of N-W generator. The selection procedure can be represented as a CNF/DNF, as the seed length for Ext is at most $\Theta(\log n)$. (Detailed proof is the same as the proof of Lemma 4.10.)

As a result, we need a circuit of depth $a+2$ to realize Samp.

For the special situation that $c = 0$, the seed length d for Ext can be $\Theta(a \log\log n)$. So $t_0 = 2^d = (\log n)^{\Theta(a)}$. □

After sampling, we give an extractor with smaller errors that can be applied on the samples. Specifically, we use leftover hash lemma.

Lemma 5.19 (Leftover Hash Lemma [26])**.** *Let X be an $(n', k = \delta n')$-source. For any $\Delta > 0$, let H be a universal family of hash functions mapping n' bits to $m = k - 2\Delta$ bits. The distribution $U \circ \mathsf{Ext}(X, U)$ is at distance at most $1/2^{\Delta}$ to uniform distribution where the function $\mathsf{Ext} : \{0,1\}^{n'} \times \{0,1\}^d \to \{0,1\}^m$ chooses the U'th hash function h_U in H and outputs $h_U(X)$.*

We use the following universal hash function family $H = \{h_u, u \in \{0,1\}^{n'}\}$. For every u, the hash function $h_u(x)$ equals to the last m bits of $u \cdot x$ where $u \cdot x$ is computed in $\mathbb{F}_{2^{n'}}$.

Specifically, for any constant $a \in \mathbb{N}^+$, for any $n' = \Theta(\log^a n)$ then Ext can be computed by an AC^0 circuit of depth $a+1$.

Proof. The proof in [26] has already shown that the universal hash function is a strong extractor. We only need to show that the hash functions can be computed in AC^0.

Given a seed u, we need to compute $u \cdot x$ which is a multiplication in $\mathbb{F}_{2^{n'}}$. We claim that this can be done in AC^0. Note that since the multiplication is in $\mathbb{F}_{2^{n'}}$, it is also a bi-linear function when regarding the two inputs as two n'-bit strings. Thus, each output bit is essentially the inner product over some input bits.

This shows that each output bit of $p \cdot q$ is an inner product of two vectors of n' dimension. As $n' = \Theta(\log^a n)$, by Lemma 2.11, this can be done in AC^0 of depth $a+1$ and size $\mathsf{poly}(n)$. All the output bits can be computed in parallel. So $u \cdot x$ can be computed in AC^0 of depth $a+1$ and size $\mathsf{poly}(n)$.

□

Theorem 5.20. *For any constant $a \in \mathbb{N}^+$, any constant $\delta \in (0,1]$ and any $\epsilon = 1/2^{\Theta(\log^a n)}$, there exists an explicit construction of a $(k = \delta n, \epsilon)$-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ in AC^0 of depth $\Theta(a)$, where $d = (\log n)^{\Theta(a)}$, $m = \Theta(\log^a n)$ and the locality is $(\log n)^{\Theta(a)}$.*

Proof. We follow the sample-then-extract procedure.

Let $\mathsf{Samp} : \{0,1\}^{r_s} \to \{0,1\}^t$ be a (μ_1, μ_2, γ)-averaging sampler following from Lemma 5.18. Let $\tau = 0.1\delta$, $\mu_1 = (\delta - 2\tau)/\log(1/\tau), \mu_2 = (\delta - 3\tau)/\log(1/\tau)$, $\gamma = 0.8\epsilon$. As a result, μ_1 is a constant and $\mu_2 = \alpha\mu_1$ for some constant $\alpha \in (0,1)$. For an (n,k)-source X, by Lemma 5.15, we have $\mathsf{SD}(R \circ X_{\mathsf{Samp}(R)}, R \circ W) \le \gamma + 2^{-\Omega(\tau n)}$. Here R is a uniform random variable. For every r in $\{0,1\}^{r_s}$, the random variable $W|_{R=r}$ is a $(t, (\delta - 3\tau)t)$-source.

By Lemma 5.18, $r_s = \Theta(\log^a n)$ and t can be any integer in $[t_0, n]$ with $t_0 = (\log n)^{\Theta(a)}$.

Let $m = 6\log^a n$, $\epsilon_1 = 0.1\epsilon$. Let t be such that $(\delta - 3\tau)t \geq m + 2\log(1/\epsilon_1)$. Let $\mathsf{Ext}_1 : \{0,1\}^t \times \{0,1\}^{d_1} \to \{0,1\}^m$ be a $((\delta - 3\tau)t, \epsilon_1)$-extractor following from Lemma 5.19. As a result,

$$\mathsf{SD}(U \circ \mathsf{Ext}_1(W, U), U') \leq \epsilon_1,$$

where U, U' are uniform distributions.

As a result, the sample-then-extract procedure gives an extractor of error

$$\gamma + 2^{-\Omega(\tau n)} + \epsilon_1 \leq 0.8\epsilon + 2^{-\Omega(\tau n)} + 0.1\epsilon.$$

As τ is a constant, $2^{-\Omega(\tau n)} \leq 0.1\epsilon$.

Thus the error of the extractor is at most ϵ.

The seed length is $r_s + d_1 = \Theta(\log^a n + t) = (\log n)^{\Theta(a)}$.

The locality is $t = (\log n)^{\Theta(a)}$ because when the seed is fixed, we select t bits from X by sampling.

The sampler Samp is in AC^0 of depth $a + 1$. The extractor Ext_1 is in AC^0 of depth $\Theta(a)$. So Ext is in AC^0 of depth $\Theta(a)$

□

In this way, we in fact have an extractor with a smaller error comparing to Lemma 5.1.

Next we give the construction for error reduction of super-polynomially small errors.

Construction 5.21 (Error Reduction for Super-Polynomially Small Error)**.** *For any constant $a \in \mathbb{N}^+$, any constant $c \in \mathbb{N}$, any $k = \Theta(n/\log^c n)$ and any $\epsilon = 1/2^{\Theta(\log^a n)}$, let X be an (n, k)-source. We construct a strong (k, ϵ)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with $m = k^{\Omega(1)}$.*

- *Let $\mathsf{Ext}_0 : \{0,1\}^{n_0 = n} \times \{0,1\}^{d_0} \to \{0,1\}^{m_0}$ be a (k_0, ϵ_0)-extractor following from Theorem 4.11 with $k_0 \leq k - \Delta_1, \Delta_1 = \log(n/\epsilon)$, $\epsilon_0 = k^{-\Theta(1)}$, $d_0 = \Theta(\log n)$, $m_0 = k^{O(1)}$.*
- *Let $\mathsf{Ext}_1 : \{0,1\}^{n_1 = m_0/t_2} \times \{0,1\}^{d_1} \to \{0,1\}^{m_1}$ be a (k_1, ϵ_1)-extractor following from Theorem 5.20 where $k_1 = 0.9n_1$, $\epsilon_1 = \epsilon/n$, $d_1 = (\log n)^{\Theta(a)}$, $m_1 = (\log n)^{\Theta(a)}$.*
- *Let t_1 be such that $(2\epsilon_0)^{t_1} \leq 0.1\epsilon$. (We only consider the case that $\epsilon < \epsilon_0$. If $\epsilon \geq \epsilon_0$, we set Ext to be Ext_0.)*
- *Let $t_2 = m_0^{1/3}$.*

Our construction is as follows.

1. *Let $R_1, R_2, \ldots, R_{t_1}$ be independent uniform distributions such that for every $i \in [t_1]$ the length of R_i is d_0. Get $Y_1 = \mathsf{Ext}_0(X, R_1), \ldots, Y_{t_1} = \mathsf{Ext}_0(X, R_{t_1})$.*
2. *Get $Y = Y_1 \circ Y_2 \circ Y_3 \circ \cdots \circ Y_{t_1}$.*
3. *For each $i \in [t_1]$, let $Y_i = Y_{i,1} \circ Y_{i,2} \circ \cdots \circ Y_{i,t_2}$ such that for every $j \in [t_2]$, $Y_{i,j}$ has length $n_1 = m_0/t_2$. Let $S_1, S_2, \ldots, S_{t_1}$ be independent uniform distributions, each having length d_1. Get $Z_{i,j} = \mathsf{Ext}_1(Y_{i,j}, S_i), \forall i \in [t_1], j \in [t_2]$. Let $Z_i = Z_{i,1} \circ Z_{i,2} \circ \cdots Z_{i,t_2}$.*
4. *Let $R = \bigcirc_i R_i, S = \bigcirc_i S_i$. We get $\mathsf{Ext}(X, U) = Z = \bigoplus_i^{t_1} Z_i$ where $U = R \circ S$.*

Theorem 5.22. *For any constant $a \in \mathbb{N}^+$, any constant $c \in \mathbb{N}$, any $k = \Theta(n/\log^c n)$, any $\epsilon = 1/2^{\Theta(\log^a n)}$ and any constant $\gamma \in (0,1)$, there exists an explicit construction of a strong (k, ϵ)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ in AC^0 of depth $\Theta(a + c)$, where $d = (\log n)^{\Theta(a)}$, $m = n^{1/10800}(\log n)^{\Theta(a)} = k^{\Omega(1)}$ and the locality is $(\log n)^{\Theta(a+c)}$.*

Proof Sketch. The proof is almost the same as that of Theorem 5.13. We only need to make sure the parameters are correct.

There are two differences between Construction 5.21 and Construction 5.6. First, in Construction 5.21, the extractor Ext_1 follows from Lemma 5.18. Second, the ϵ in Construction 5.21 is super-polynomially small.

We first claim that Ext is a (k, ϵ)-extractor. The proof strategy is the same as that of Lemma 5.7, except that the parameters need to be modified.

As $(2\epsilon_0)^{t_1} \leq 0.1\epsilon$, we can take $t_1 = \Theta(\log^{a-1} n)$. According to the proof of Lemma 5.7, the error of Ext is

$$(2\epsilon_0)^{t_1} + (2^{-\Delta_1} + \epsilon' + (\epsilon'_0 + \epsilon_1)t_2)$$

Here ϵ', ϵ'_0 follow the same definitions as those in the proof of Lemma 5.7. As a result, we know that both ϵ' and ϵ'_0 can be $1/2^{k^{\Omega(1)}}$.

Also we know that $\Delta_1 = \log(n/\epsilon)$, $\epsilon_1 = \epsilon/n$, $t_2 = k^{O(1)}$.

So we can set ϵ', ϵ'_0 small enough so that $2^{-\Delta_1} + \epsilon' + (\epsilon'_0 + \epsilon_1)t_2 \leq 0.1\epsilon$.

As a result, the overall error will be at most ϵ.

By the same proof as that of Lemma 5.10, we know that the output length is

$$m = t_2 \times m_1 = m_0^{1/3}(\log n)^{\Theta(a)} = k^{\Omega(1)}.$$

For the seed length, we know that according to the settings of Ext_0 and Ext_1, as $t_1 = \Theta(\log^{a-1} n)$, $d = O(t_1 d_0 + t_1 d_1) = (\log n)^{\Theta(a)}$.

As the locality of Ext_0 is $l_0 = (\log n)^{\Theta(c)}$, and the locality of Ext_1 is $l_1 = (\log n)^{\Theta(a)}$, the locality of Ext is $t_1 \times l_1 \times l_0 = (\log n)^{\Theta(a+c)}$.

According to our settings we know that Ext_0 is in AC^0 of depth $\Theta(c)$ and Ext_0 is in AC^0 of depth $\Theta(a)$. Also we know that $t_1 = \Theta(\log^{a-1} n), t_2 = m_0^{1/3}$, so the XOR of t_1 bits can be computed in AC^0 of depth $\Theta(a)$ and size $\mathsf{poly}(n)$ by Lemma 2.11. So Ext is in AC^0 of depth $\Theta(a + c)$.

□

6 Output Length Optimization for AC^0 Extractors

In this section, we show how to extract $(1 - \gamma)k$ bits for any constant $\gamma > 0$.

6.1 Output Length OPT for Polynomially Small Error

By Theorem 5.13, we have a (k, ϵ)-extractor in AC^0 for any $k = n/\mathsf{poly}(\log n)$ and any $\epsilon = 1/\mathsf{poly}(n)$.

According to Lemma 5.17, we can have the following lemma which gives an explicit construction of oblivious samplers and averaging samplers.

Lemma 6.1. *For any $\gamma = 1/\mathsf{poly}(n)$ and any $\epsilon = 1/\mathsf{poly}(\log n)$, there exists an explicit $(O(\log n), \log n, t, \gamma, \epsilon)$-oblivious sampler for any integer t in $[t_0, n]$ with $t_0 = \mathsf{poly}(\log n)$.*

Let $\alpha \in (0, 1)$ be an arbitrary constant. For any $\mu = 1/\mathsf{poly}(\log n)$ and any $\gamma = 1/\mathsf{poly}(n)$, there exists an explicit $(\mu, \alpha\mu, \gamma)$-averaging sampler $\mathsf{Samp} : \{0,1\}^{O(\log n)} \to [n]^t$, for any integer t in $[t_0, n]$ with $t_0 = \mathsf{poly}(\log n)$.

Proof. Let $k = 2\log n$. Consider a (k, ϵ)-extractor $\mathsf{Ext} : \{0,1\}^{c \log n} \times \{0,1\}^d \to \{0,1\}^{\log n}$ for some constant $c \in \mathbb{N}^+$, following Lemma 2.7. Here we make one modification. We replace the last d bits of the output with the seed. We can see in this way, Ext is still an extractor.

The entropy rate is $\delta = 2/c$. By Lemma 2.7, we know that, d can be $\Theta(\log(c\log n) + \log(1/\epsilon)) = \Theta(\log\log n)$.

As a result, by Lemma 5.17, there exists an explicit $(c\log n, \log n, t, \gamma, \epsilon)$-oblivious sampler where $t = 2^d = \mathsf{poly}(\log n)$, $\gamma = 2^{1-(1-2/c)(c\log n)}$, $\epsilon = 1/\mathsf{poly}(\log n)$. For t, we claim that t can be any number in the range $[t_0, n]$ with $t_0 = \mathsf{poly}(\log n)$. This is because we can always add more bits in the seed (The total length of the seed can be added up to $\log n$). As we do not require the extractor to be strong, we can always use the seed to replace the last d bits of the output. This shows that t can be any number in $[t_0, n]$ with $t_0 = \mathsf{poly}(\log n)$. Since c can be any large enough constant, γ can be any $1/\mathsf{poly}(n)$.

According to the definition of the oblivious sampler, we know that $\forall f : [n] \to [0,1]$,

$$\Pr_{I \leftarrow \mathsf{Samp}(U)}[|\frac{1}{t}\sum_{i\in I} f(i) - \mathbf{E}f| > \epsilon] \le \gamma.$$

Next we consider the definition of the averaging sampler.

Let $(1-\alpha)\mu = \epsilon$. As $\mu = 1/\mathsf{poly}(\log n)$, $\epsilon = \mathsf{poly}(\log n)$. For any $f : [n] \to [0,1]$ such that $\mu \le \mathbf{E}f$, we have the following inequalities.

$$\begin{aligned}
&\Pr_{I \leftarrow \mathsf{Samp}(U)}[\frac{1}{t}\sum_{i\in I} f(i) < \alpha\mu] \\
=&\Pr_{I \leftarrow \mathsf{Samp}(U)}[\frac{1}{t}\sum_{i\in I} f(i) < \mu - \epsilon] \\
=&\Pr_{I \leftarrow \mathsf{Samp}(U)}[\mu - \frac{1}{t}\sum_{i\in I} f(i) > \epsilon] \\
\le&\Pr_{I \leftarrow \mathsf{Samp}(U)}[\mathbf{E}f - \frac{1}{t}\sum_{i\in I} f(i) > \epsilon] \\
\le&\Pr_{I \leftarrow \mathsf{Samp}(U)}[|\frac{1}{t}\sum_{i\in I} f(i) - \mathbf{E}f| > \epsilon] \\
\le&\gamma.
\end{aligned} \tag{17}$$

The first inequality holds because if the event that $\mu - \frac{1}{t}\sum_{i\in I} f(i) > \epsilon$ happens, then the event that $\mathbf{E}f - \frac{1}{t}\sum_{i\in I} f(i) > \epsilon$ must happen, as $\mu \le \mathbf{E}f$. The second inequality is because $\mathbf{E}f - \frac{1}{t}\sum_{i\in I} f(i) \le |\mathbf{E}f - \frac{1}{t}\sum_{i\in I} f(i)|$. So if $\mathbf{E}f - \frac{1}{t}\sum_{i\in I} f(i) > \epsilon$ happens, then $|\frac{1}{t}\sum_{i\in I} f(i) - \mathbf{E}f| > \epsilon$ happens.

Also as we replace the last d bits of the output of our extractor with the seed, the samples are distinct according to the construction of Lemma 5.17.

This meets the definition of the averaging sampler. So this also gives an explicit $(\mu, \alpha\mu, \gamma)$-averaging sampler.

□

By Lemma 5.15, we can sample several times to get a block source.

Lemma 6.2 (Sample a Block Source)**.** *Let t be any constant in $\mathbb{N}^+$. For any $\delta > 0$, let X be an $(n, k = \delta n)$-source . Let $\mathsf{Samp} : \{0,1\}^r \to [n]^m$ be a (μ_1, μ_2, γ)-averaging sampler where $\mu_1 = (\frac{1}{t}\delta - 2\tau)/\log(1/\tau)$ and $\mu_2 = (\frac{1}{t}\delta - 3\tau)/\log(1/\tau)$, $m = (\frac{t-1}{t}k - \log(1/\epsilon_0))/t$. Let $\epsilon_s = \gamma + 2^{-\Omega(\tau n)}$. For any $i \in [t]$, let U_is be uniform distributions over $\{0,1\}^r$. Let $X_i = X_{\mathsf{Samp}(U_i)}$, for $i \in [t]$.*

It concludes that $\bigcirc_{i=1}^t U_i \circ \bigcirc_{i=1}^t X_i$ is $\epsilon = t(\epsilon_s + \epsilon_0)$-close to $\bigcirc_{i=1}^t U_i \circ \bigcirc_{i=1}^t W_i$ where for every $u \in \mathsf{supp}(\bigcirc_{i=1}^t U_i)$, conditioned on $\bigcirc_{i=1}^t U_i = u$, $\bigcirc_{i=1}^t W_i$ is a $(k_1, k_2, \ldots, k_t)$-block source with block size m and $k_1 = k_2 = \cdots = k_t = (\delta/t - 3\tau)m$. Here ϵ_0 can be as small as $1/2^{\Omega(k)}$.

Proof. We prove by induction on $i \in [t]$.

If $i = 1$, according to Lemma 5.15, we know $U_1 \circ X_1$ is $\epsilon_s = (\gamma + 2^{-\Omega(\tau n)})$-close to $U_1 \circ W$ such that $\forall u \in \mathsf{supp}(U_1), H_\infty(W|_{U_1=u}) = (\delta/t - 3\tau)m$.

Next we prove the induction step.

Suppose $\bigcirc_{j=1}^{i} U_j \circ \bigcirc_{j=1}^{i} X_j$ is $(\epsilon_s + \epsilon_0)i$-close to $\bigcirc_{j=1}^{i} U_j \circ \bigcirc_{j=1}^{i} W_j$, where for every $u \in \{0,1\}^{ir}$, conditioned on $\bigcirc_{j=1}^{i} U_j = u$, $\bigcirc_{j=1}^{i} W_j$ is a $(k_1, k_2, \ldots, k_i)$-block source with block size m and $k_1 = k_2 = \cdots = k_i = (\delta/t - 3\tau)m$.

Consider $i+1$. Recall the Chain Rule Lemma 5.8. First notice that $\bigcirc_{j=1}^{i} U_j \circ \bigcirc_{j=1}^{i} X_j \circ X$ has entropy $ir + k$. Then we know that $\bigcirc_{j=1}^{i} U_j \circ \bigcirc_{j=1}^{i} X_j \circ X$ is ϵ_0-close to $\bigcirc_{j=1}^{i} U_j \circ \bigcirc_{j=1}^{i} X_j \circ X'$ such that for every $u \in \{0,1\}^{ir}$ and every $x \in \{0,1\}^{im}$, conditioned on $\bigcirc_{j=1}^{i} U_j = u, \bigcirc_{j=1}^{i} X_j = x$, X' has entropy $k - im - \log(1/\epsilon_0) \geq k/t$ which means the entropy rate is at least δ/t.

By our assumption for i, $\bigcirc_{j=1}^{i} U_j \circ \bigcirc_{j=1}^{i} X_j \circ X'$ is $(\epsilon_s + \epsilon_0)i$-close to $\bigcirc_{j=1}^{i} U_j \circ \bigcirc_{j=1}^{i} W_j \circ \tilde{X}$, where $\tilde{X}$ is a random variable such that $\forall u \in \{0,1\}^{ir}, \forall x \in \{0,1\}^{im}$, $\tilde{X}|_{\bigcirc_{j=1}^{i} U_j = u, \bigcirc_{j=1}^{i} X_j = x}$ has the same distribution as $X'|_{\bigcirc_{j=1}^{i} U_j = u, \bigcirc_{j=1}^{i} W_j = x}$. As a result, for every $u \in \{0,1\}^{ir}$ and $x \in \{0,1\}^{im}$, conditioned on $\bigcirc_{j=1}^{i} U_j = u, \bigcirc_{j=1}^{i} W_j = x$, $\tilde{X}$ has entropy $k - im - \log(1/\epsilon_0) \geq k/t$.

Denote the event $(\bigcirc_{j=1}^{i} U_j = u, \bigcirc_{j=1}^{i} W_j = x)$ as e, by Lemma 5.15, by sampling on source $\tilde{X}|_e$, we get $U_{i+1} \circ (\tilde{X}|_e)_{\mathsf{Samp}(U_{i+1})} = U_{i+1} \circ \tilde{X}_{\mathsf{Samp}(U_{i+1})}|_e$. It is ϵ_s-close to $U_{i+1} \circ W|_e$ where $\forall a \in \{0,1\}^r$, $(W|_e)|_{U_{i+1}=a}$ is a $(m, (\delta/t - 3\tau)m)$-source. Thus $\bigcirc_{j=1}^{i+1} U_j \circ \bigcirc_{j=1}^{i} W_j \circ \tilde{X}_{\mathsf{Samp}(U_{i+1})}$ is ϵ_s-close to $\bigcirc_{j=1}^{i+1} U_j \circ \bigcirc_{j=1}^{i} W_j \circ W$.

Let $W_{i+1} = W$. As a result, $\bigcirc_{j=1}^{i+1} U_j \circ \bigcirc_{j=1}^{i} X_j$ is $(\epsilon_s + \epsilon_0)(i+1)$-close to $\bigcirc_{j=1}^{i+1} U_j \circ \bigcirc_{j=1}^{i+1} W_j$ such that for every $u \in \{0,1\}^{ir}$, conditioned on $\bigcirc_{j=1}^{i+1} U_j = u$, $\bigcirc_{j=1}^{i+1} W_j$ is a $(k_1, k_2, \ldots, k_i)$-block source with block size m and $k_1 = k_2 = \cdots = k_{i+1} = (\delta/t - 3\tau)m$.

This proves that induction step.

□

This lemma reveals a way to get a block source by sampling. Block sources are easier to extract.

Another important technique is the parallel extraction. According to Raz at al. [40], we have the following lemma.

Lemma 6.3 ([40])**.** *Let* $\mathsf{Ext}_1 : \{0,1\}^n \times \{0,1\}^{d_1} \to \{0,1\}^{m_1}$ *be a strong* (k, ϵ)*-extractor with entropy loss* Δ_1 *and* $\mathsf{Ext}_2 : \{0,1\}^n \times \{0,1\}^{d_2} \to \{0,1\}^{m_2}$ *be a strong* $(\Delta_1 - s, \epsilon_2)$*-extractor with entropy loss* Δ_2 *for any* $s < \Delta_1$*. Suppose the function* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^{d_1+d_2} \to \{0,1\}^{m_1+m_2}$ *is as follows.*

$$\mathsf{Ext}(x, u_1 \circ u_2) = \mathsf{Ext}_1(x, u_1) \circ \mathsf{Ext}_2(x, u_2)$$

Then Ext *is a strong* $(k, (\frac{1}{1-2^{-s}})\epsilon_1 + \epsilon_2 \leq \epsilon_1 + \epsilon_2 + 2^{-s})$*-extractor with entropy loss* $\Delta_2 + s$*.*

This can be generalized to the parallel extraction for multiple times.

Lemma 6.4. *Let* X *be an* (n, k)*-source. Let* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *be a strong* (k_0, ϵ)*-extractor with* $k_0 = k - tm - s$ *for any* t, s *such that* $tm + s < k$*. Let* $\mathsf{Ext}' : \{0,1\}^n \times \{0,1\}^{td} \to \{0,1\}^{tm}$ *be constructed as follows.*

$$\mathsf{Ext}'(x, \bigcirc_{i=1}^{t} u_i) = \mathsf{Ext}(x, u_1) \circ \mathsf{Ext}(x, u_2) \circ \cdots \circ \mathsf{Ext}(x, u_t)$$

Then Ext' *is a strong* $(k, t(\epsilon + 2^{-s}))$*-extractor.*

Proof. Consider the mathematical induction on j.

For $j = 1$, it is true. As Ext is a strong (k_0, ϵ)-extractor, it is also a strong $(k, j(\epsilon + 2^{-s}))$-extractor.

Next we prove the induction step.

Assume it is true for j. Consider $j + 1$.

$$\mathsf{Ext}'(x, \bigcirc_{i=1}^{j+1} u_i) = \mathsf{Ext}'(x, \bigcirc_{i=1}^{j} u_i) \circ \mathsf{Ext}(x, u_{j+1})$$

Here $\mathsf{Ext}'(x, \bigcirc_{i=1}^{j} u_i)$ is a strong $(k, j(\epsilon + 2^{-s}))$-source. Its entropy loss is $k - jm$. Also we know that Ext is a strong $(k - tm - s, \epsilon)$-extractor, thus a strong $(k - jm - s, \epsilon)$-extractor. According to Lemma 6.3, $\mathsf{Ext}'(x, \bigcirc_{i=1}^{j+1} u_i)$ is a strong $(k, (j+1)(\epsilon + 2^{-s}))$-extractor. Its entropy loss is $k - (j+1)m$.

This completes the proof.

□

Lemma 6.4 shows a way to extract more bits. Assume we have an (n, k)-source and an extractor, if the output length of the extractor is $k^\beta, \beta < 1$, then we can extract several times to get a longer output. However, if we merely do it in this way, we need a longer seed. In fact, if we extract enough times to make the output length to be $\Theta(k)$, we need a seed with length $\Theta(k^{1-\beta} \log n)$. This immediately gives us the following theorem.

Theorem 6.5. *For any constant $c \in \mathbb{N}$, any $k = \Theta(n/\log^c n)$, any $\epsilon = 1/\mathsf{poly}(n)$ and any constant $\gamma \in (0, 1)$, there exists an explicit construction of a strong (k, ϵ)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ in AC^0 of depth $c + 10$. The locality is $\Theta(\log^{c+5} n) = \mathsf{poly}(\log n)$. The seed length $d = O(\frac{k}{n^{1/10800}})$. The output length $m = (1 - \gamma)k$.*

Proof. Let $\mathsf{Ext}_0 : \{0,1\}^{n_0} \times \{0,1\}^{d_0} \to \{0,1\}^{m_0}$ be a $(k_0, \epsilon_0 = \epsilon/n)$ extractor following from Theorem 5.13. Here $k_0 = k - tm_0 - s$ where $s = \log(n/\epsilon)$, $t = (1 - \gamma)k/m_0$. By lemma 6.4, we know that there exists a (k, ϵ') extractor Ext with $\epsilon' = t(\epsilon_0 + 2^{-s}) \le \epsilon$. The output length is $(1 - \gamma)k$.

According to the construction in Lemma 6.4, Ext has the same circuit depth as Ext_0. So Ext is in AC^0 of depth $c + 10$. The locality of Ext is also the same as that of Ext_0 which is $\Theta(\log^{c+5} n) = \mathsf{poly}(\log n)$. The seed length is $t \times O(\log n) = O(\frac{k}{n^{1/10800}})$.

□

In order to achieve a small seed length, next we use classic bootstrapping techniques to extract more bits. Our construction will be in AC^0. However, our construction cannot keep the locality small.

Construction 6.6. *For any $c \in \mathbb{N}$, any $k = \delta n = \Theta(n/\log^c n)$ and any $\epsilon = 1/\mathsf{poly}(n)$, we construct a (k, ϵ)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ where $d = O(\log n), m = \Theta(\delta k)$.*

- *Let X be an (n, k)-source*
- *Let $t \ge 10800$ be a constant.*
- *Let $\mathsf{Samp} : \{0,1\}^r \to [n]^{m_s}$ be a (μ_1, μ_2, γ)-averaging sampler following from Lemma 6.1, where $\mu_1 = (\frac{1}{t}\delta - 2\tau)/\log(1/\tau)$ and $\mu_2 = (\frac{1}{t}\delta - 3\tau)/\log(1/\tau)$, $m_s = (\frac{t-1}{t}k - \log(1/\epsilon_0))/t$, $\tau = \frac{1}{4}\delta$, $\gamma = \epsilon/n$. Let $\epsilon_s = \gamma + 2^{-\Omega(\tau n)}$.*
- *Let $\mathsf{Ext}_0 : \{0,1\}^{n_0 = m_s} \times \{0,1\}^{d_0} \to \{0,1\}^{m_0}$ be a (k_0, ϵ_0)-extractor following from Theorem 5.13 where $k_0 = 0.1(\frac{1}{t}\delta - 3\tau)m_s - s$, $\epsilon_0 = \epsilon/(10tn)$, $d_0 = O(\log n_0)$, $m_0 = \Theta(n_0^{1/10800} \log n_0)$. Let s be such that $2^{-s} \le \epsilon/(10tn)$.*

Next we construct the function Ext as follows.

1. *Get Let* $X_i = X_{\mathsf{Samp}(S_i)}$ *for* $i \in [t]$*, where* $S_i, i \in [t]$ *are independent uniform distributions.*
2. *Get* $Y_t = \mathsf{Ext}_0(X_t, U_0)$ *where* U_0 *is the uniform distribution with length* d_0*.*
3. *For* $i = t-1$ *to* 1*, get* $Y_i = \mathsf{Ext}'(X_i, Y_{i+1})$ *sequentially.*
4. *Output* $\mathsf{Ext}(X, U_d) = Y_1 = \mathsf{Ext}'(X_1, Y_2)$*, where* $U_d = U_0 \circ \bigcirc_{i=1}^{t} S_i$ *and the function* Ext' *is defined as follows.*

$$\mathsf{Ext}'(x, r) = \bigcirc_{i=1}^{\min\{\lfloor |r|/d_0 \rfloor, \lfloor 0.9(\frac{1}{t}\delta - 3\tau)m_s/m_0 \rfloor\}} \mathsf{Ext}_0(x, r_i),$$

where $r = \bigcirc_{i=1}^{\lfloor |r|/d_0 \rfloor} r_i \circ r'$ *for some extra bits* r' *and* $\forall i, |r_i| = d_0$*.*

Lemma 6.7. *For* $\epsilon_1 = 1/2^{\Omega(k)}$ *and for any* $\epsilon_s = 1/\mathsf{poly}(n)$*,* $\bigcirc_{i=1}^{t} S_i \circ \bigcirc_{i=1}^{t} X_i$ *is* $t(\epsilon_s + \epsilon_1)$*-close to* $\bigcirc_{i=1}^{t} S_i \circ \bigcirc_{i=1}^{t} W_i$ *where* S_i*s are independent uniform distributions.*

Here $\forall r \in \mathsf{supp}(\bigcirc_{i=1}^{t} S_i)$*, conditioned on* $\bigcirc_{i=1}^{t} S_i = r$*,* $\bigcirc_{i=1}^{t} W_i$ *is a* $(k_1, k_2, \ldots, k_t)$*-block source with* $k_1 = k_2 = \cdots = k_t = k' = (\frac{1}{t}\delta - 3\tau)m_s$ *.*

Proof. It follows from Lemma 6.2.

□

Lemma 6.8. *In Construction 6.6, the function* Ext *is a strong* $(k, \tilde{\epsilon})$*-extractor with*

$$\tilde{\epsilon} = t(\epsilon_s + \epsilon_1) + tk(\epsilon_0 + 2^{-s}).$$

By setting $\epsilon_1 = 1/2^{\Omega(k)}$*, according to the settings of* ϵ_s, ϵ_0 *and* s*, we have* $\tilde{\epsilon} \le \epsilon$*.*

Proof of Lemma 6.8. By Lemma 6.7, $\bigcirc_{i=1}^{t} S_i \circ \bigcirc_{i=1}^{t} X_i$ is $t(\epsilon_s + \epsilon_1) = 1/\mathsf{poly}(n)$-close to $\bigcirc_{i=1}^{t} S_i \circ B$ where $B = B_1 \circ B_2 \circ \ldots \circ B_t$. The S_is are independent uniform distributions. Also $\forall s \in \mathsf{supp}(\bigcirc_{i=1}^{t} S_i)$, conditioned on $\bigcirc_{i=1}^{t} S_i = s$, B is a $(k_1, k_2, \ldots, k_t)$-block source with $k_1 = k_2 = \cdots = k_t = k' = (\frac{1}{t}\delta - 3\tau)m_s$. We denote the first i blocks to be $\tilde{B}_i = \bigcirc_{j=1}^{i} B_i$.

Let $Y'_i = \mathsf{Ext}'(B_i, Y'_{i+1})$ for $i = 1, 2, \ldots, t$ where $Y'_{t+1} = U_0$ is the uniform distribution with length d_0.

Next we use induction over i (from t to 1) to show that

$$\mathsf{SD}(U_0 \circ Y'_i, U) \le (t + 1 - i)k(\epsilon_0 + 2^{-s}).$$

The basic step is to prove that $\forall b_1, b_2, \ldots, b_{t-1} \in \{0,1\}^{m_s}$, conditioned on $B_1 = b_1, \ldots, B_{t-1} = b_{t-1}$, $\mathsf{SD}(U_0 \circ Y'_t, U) \le k(\epsilon_0 + 2^{-s})$. According to the definition of Ext',

$$\mathsf{SD}(U_0 \circ \mathsf{Ext}'(B_t, U_0), U) \le \epsilon_0.$$

This proves the basic step.

For the induction step, assume that $\forall b_1, b_2, \ldots, b_{i-1} \in \{0,1\}^{m_s}$, conditioned on $B_1 = b_1, \ldots, B_{i-1} = b_{i-1}$,

$$\mathsf{SD}(U_0 \circ Y'_i, U) \le (t + 1 - i)k(\epsilon_0 + 2^{-s}).$$

Consider $U_0 \circ Y'_{i-1} = U_0 \circ \mathsf{Ext}'(B_{i-1}, Y'_i)$.

We know that $\forall b_1, b_2, \ldots, b_{t-2} \in \{0,1\}^{m_s}$, conditioned on $B_1 = b_1, \ldots, B_{i-2} = b_{i-2}$, $\tilde{B}_{i-1} \circ U_0 \circ Y'_i$ is a convex combination of $b_{i-1} \circ U_0 \circ Y'_i, \forall b_{i-1} \in \mathsf{supp}(\tilde{B}_{i-1})$. As a result,

$$\mathsf{SD}(\tilde{B}_{i-1} \circ U_0 \circ Y'_i, \tilde{B}_{i-1} \circ U) \le (t + 1 - i)k(\epsilon_0 + 2^{-s}).$$

Thus, $\forall b_1, b_2, \ldots, b_{t-2} \in \{0,1\}^{m_s}$, conditioned on $B_1 = b_1, \ldots, B_{i-2} = b_{i-2}$, as $\tilde{B}_{i-1} \circ U_0 \circ Y_i'$ is a convex combination of $b_{i-1} \circ U_0 \circ Y_i', \forall b_{i-1} \in \mathsf{supp}(\tilde{B}_{i-1})$ and $\tilde{B}_{i-1} \circ U$ is a convex combination of $b_{i-1} \circ U, \forall b \in \mathsf{supp}(\tilde{B}_{i-1})$, by Lemma 2.4 part 2,

$$\begin{aligned}
&\mathsf{SD}(U_0 \circ \mathsf{Ext}'(B_{i-1}, Y_i'), U_1 \circ \mathsf{Ext}'(B_{i-1}, U_2)) \\
\leq &\mathsf{SD}(\tilde{B}_{i-1} \circ U_0 \circ Y_i', \tilde{B}_{i-1} \circ U) \\
\leq &(t+1-i)k(\epsilon_0 + 2^{-s}).
\end{aligned} \tag{18}$$

Here $U = U_1 \circ U_2$. U_1 is the uniform distribution having $|U_1| = |U_0|$. U_2 is the uniform distribution having $|U_2| = |Y_i'|$.

According to the definition of Ext' and Lemma 6.4, we know that $\forall b_1, b_2, \ldots, b_{i-2} \in \{0,1\}^{m_s}$, conditioned on $B_1 = b_1, \ldots, B_{i-2} = b_{i-2}$,

$$\mathsf{SD}(U_1 \circ \mathsf{Ext}'(B_{i-1}, U_2), U) \leq k(\epsilon_0 + 2^{-s}).$$

So according to triangle inequality of Lemma 2.4, $\forall b_1, b_2, \ldots, b_{t-2} \in \{0,1\}^{m_s}$, conditioned on $B_1 = b_1, \ldots, B_{i-2} = b_{i-2}$,

$$\begin{aligned}
&\mathsf{SD}(U_0 \circ Y_{i-1}', U) \\
= &\mathsf{SD}(U_0 \circ \mathsf{Ext}'(B_{i-1}, Y_i'), U) \\
\leq &\mathsf{SD}(U_0 \circ \mathsf{Ext}'(B_{i-1}, Y_i'), U_1 \circ \mathsf{Ext}'(B_{i-1}, U_2)) + \mathsf{SD}(U_1 \circ \mathsf{Ext}'(B_{i-1}, U_2), U) \\
\leq &(t+1-i)k(\epsilon_0 + 2^{-s}) + k(\epsilon_0 + 2^{-s}) \\
= &(t+1-(i-1))k(\epsilon_0 + 2^{-s}).
\end{aligned} \tag{19}$$

This proves the induction step.

So we have $\mathsf{SD}(U_0 \circ Y_1', U) \leq tk(\epsilon_0 + 2^{-s})$.

As a result,

$$\begin{aligned}
&\mathsf{SD}(U_d \circ \mathsf{Ext}(X, U_d), U) \\
= &\mathsf{SD}(U_0 \circ \bigcirc_{i=1}^{t} S_i \circ Y_1, U) \\
\leq &\mathsf{SD}(U_0 \circ \bigcirc_{i=1}^{t} S_i \circ Y_1, U_0 \circ \bigcirc_{i=1}^{t} S_i \circ Y_1') + \mathsf{SD}(U_0 \circ \bigcirc_{i=1}^{t} S_i \circ Y_1', U) \\
\leq &\mathsf{SD}(U_0 \circ \bigcirc_{i=1}^{t} S_i \circ \bigcirc_{i=1}^{t} X_i, U_0 \circ \bigcirc_{i=1}^{t} S_i \circ \bigcirc_{i=1}^{t} B_i) + \mathsf{SD}(U_0 \circ \bigcirc_{i=1}^{t} S_i \circ Y_1', U) \\
\leq &t(\epsilon_s + \epsilon_1) + tk(\epsilon_0 + 2^{-s}).
\end{aligned} \tag{20}$$

According to the settings of $\epsilon_0, \epsilon_s, t, \epsilon_1$, we know the error is at most ϵ.

□

Lemma 6.9. *In Construction 6.6, the length of Y_i is*

$$|Y_i| = \Theta(\min\{m_0(\frac{m_0}{d_0})^{t-i}, 0.9(\frac{1}{t}\delta - 3\tau)m_s\}).$$

Specifically, $m = |Y_1| = \Theta((\frac{1}{t}\delta - 3\tau)m_s) = \Theta(\delta k)$.

Proof. For each time we compute $Y_i = \mathsf{Ext}'(X_i, Y_{i+1})$, we know $|Y_i| \leq |Y_{i+1}|(\frac{m_0}{d_0})$. Also according to the definition of Ext', $|Y_i| \leq 0.9(\frac{1}{t}\delta - 3\tau)m_s$. So $|Y_i| = \Theta(\min\{m_0(\frac{m_0}{d_0})^{t-i}, 0.9(\frac{1}{t}\delta - 3\tau)m_s\})$ for $i \in [t]$.

By Theorem 5.13, $m_0 = \Theta(n_0^{1/10800} \log n)$. Also we know that $n_0 = m_s = O(tk)$. As a result, when $t \geq 10800$, $m_0(\frac{m_0}{d_0})^{t-1} = \omega(m_s)$. As a result, $m = |Y_1| = \Theta((\frac{1}{t}\delta - 3\tau)m_s) = \Theta(\delta k)$.

□

Lemma 6.10. *In Construction 6.6, the seed length* $d = \Theta(\log n)$.

Proof. The seed for this extractor is $U_d = U_0 \circ \bigcirc_{i=1}^{t} S_i$. So $|U_d| = |U_0| + \Sigma_i^t |S_i| = \Theta(\log n) + \Theta(\log n) = \Theta(\log n)$.

□

Lemma 6.11. *In Construction 6.6, the function* Ext *is in* AC^0. *The depth of the circuit is* $10800c + 97203$.

Proof. As the seed length of Samp is $O(\log n)$, the sampling procedure is in AC^0 which can be realized by a CNF/DNF. Thus the circuit depth is 2.

As the sampling procedure gives the indices for us to select bits from X, we needs 1 level of CNF/DNF to select the bits. This also needs a circuit of depth 2.

By Theorem 5.13, Ext_0 is in AC^0 with depth $c + 10$. According to the definition of Ext', it runs Ext_0 for polynomial times in parallel, so it is also in AC^0 of depth $c + 10$.

In the first step of Construction 6.6, we do sampling by t times in parallel. As a t is a constant, this operation is in AC^0 of depth 2. Next the construction do a sequence of extractions. In step 2 and 3, we run Ext' for t times sequentially. As t is a constant and Ext' is in AC^0 with depth $c + 10$, the total depth is $t(c + 10) - t + 1$.

The last step outputs Y_1, there is no gates used here, so the depth is 0.

So the function Ext in Construction 6.6 is in AC^0 with depth $t(c+10)-t+1+2+2-2 = ct+9t+3 = 10800c + 97203$.

□

Theorem 6.12. *For any constant* $\gamma \in (0, 1)$, *any* $c \in \mathbb{N}$, *any* $k = \delta n = \Theta(n/\log^c n)$ *and any* $\epsilon = 1/\mathsf{poly}(n)$, *there exists an explicit construction of a strong* (k, ϵ)*-extractor* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *in* AC^0 *with depth* $10800c + 97203$, *where* $d = \Theta(\log n)$, $m = \Theta(\delta k)$.

Proof. By Construction 6.6, Lemma 6.8, Lemma 6.9, Lemma 6.10 , and Lemma 6.11 , the conclusion immediately follows.

□

Theorem 6.13. *For any constant* $\gamma \in (0, 1)$, *any* $c \in \mathbb{N}$, *any* $k = \delta n = \Theta(n/\log^c n)$ *and any* $\epsilon = 1/\mathsf{poly}(n)$, *there exists an explicit construction of a strong* (k, ϵ)*-extractor* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *in* AC^0 *with depth* $10800c + 97203$ *where* $d = \Theta(\log^{c+1} n) = \mathsf{poly}(\log n)$, $m = (1 - \gamma)k$.

Proof. Let the extractor following Theorem 6.12 be $\mathsf{Ext}_0 : \{0,1\}^{n_0} \times \{0,1\}^{d_0} \to \{0,1\}^{m_0}$ which is a (k_0, ϵ_0)-extractor with $n_0 = n, k_0 = \gamma k - s$. The construction of Ext is

$$\mathsf{Ext}(x, u) = \bigcirc_{i=1}^{t} \mathsf{Ext}_0(x, u_i).$$

Here t is such that $tm_0 = (1 - \gamma)k$.

By Lemma 6.12, we know that $m_0 = \Theta(\delta k)$ where $\delta = \frac{1}{\log^c n}$. So $t = O(1/\delta)$. By Lemma 6.4, if $tm_0 = (1-\gamma)k$, then Ext is a (k, ϵ)-extractor with output length $(1-\gamma)k$ and $\epsilon = t(\epsilon_0 + 2^{-s})$. As s can be any $\mathsf{poly}(\log n)$ and ϵ_0 can be any $1/\mathsf{poly}(n)$, ϵ can be any $1/\mathsf{poly}(n)$. The seed length $d = td_0$. By Theorem 6.12, $d_0 = \Theta(\log n), m_0 = \Theta(\delta k)$, so $d = \Theta(\frac{\log n}{\delta}) = \Theta(\log^{c+1} n) = \mathsf{poly}(\log n)$, $m = (1 - \gamma)k$. The circuit depth maintains the same as that in Theorem 6.12 because the extraction is conducted in parallel. □

6.2 Output Length OPT for Super-Polynomially Small Error

In this subsection, we give the method which can extract more bits while having super-polynomially small errors.

Construction 6.14 (Output Length OPT for Super-Polynomial Small Errors)**.** *For any constant $a \in \mathbb{N}^+$, any constant $c \in \mathbb{N}$, any $k = \delta n = \Theta(n/\log^c n)$ and any $\epsilon = 1/2^{\Theta(\log^a n)}$, we construct a (k, ϵ)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ where $d = (\log n)^{\Theta(a)}$, $m = (1-\gamma)k$.*

- *Let X be an $(n, k = \delta n)$-source*
- *Let $t \geq 10800$ be a constant.*
- *Let $\mathsf{Samp} : \{0,1\}^r \to [n]^{m_s}$ be a (μ_1, μ_2, γ)-averaging sampler following from Lemma 5.18, where $\mu_1 = (\frac{1}{t}\delta - 2\tau)/\log(1/\tau)$ and $\mu_2 = (\frac{1}{t}\delta - 3\tau)/\log(1/\tau)$, $m_s = (\frac{t-1}{t}k - \log(1/\epsilon_0))/t$, $\tau = \frac{1}{4}\delta$, $\gamma = \epsilon/n$. Let $\epsilon_s = \gamma + 2^{-\Omega(\tau n)}$.*
- *Let $\mathsf{Ext}_0 : \{0,1\}^{n_0 = m_s} \times \{0,1\}^{d_0} \to \{0,1\}^{m_0}$ be a (k_0, ϵ_0)-extractor following from Theorem 5.22 where $k_0 = 0.1(\frac{1}{t}\delta - 3\tau)m_s - s$, $\epsilon_0 = \epsilon/(10tn)$, $d_0 = (\log n_0)^{\Theta(a)}$, $m_0 = n_0^{1/10800}\Theta(\log^a n_0)$. Let s be such that $2^{-s} \leq \epsilon/(10tn)$.*

Next we construct the function Ext as follows.

1. *Get Let $X_i = X_{\mathsf{Samp}(X,S_i)}$ for $i \in [t]$, where $S_i, i \in [t]$ are independent uniform distributions.*
2. *Get $Y_t = \mathsf{Ext}_0(X_t, U_0)$ where U_0 is the uniform distribution with length d_0.*
3. *For $i = t-1$ to 1, get $Y_i = \mathsf{Ext}'(X_i, Y_{i+1})$ sequentially.*
4. *Output $\mathsf{Ext}(X, U_d) = Y_1 = \mathsf{Ext}'(X_1, Y_2)$, where $U_d = U_0 \circ \bigcirc_{i=1}^{t} S_i$ and the function Ext' is defined as follows.*

$$\mathsf{Ext}'(x, r) = \bigcirc_{i=1}^{\min\{\lfloor |r|/d_0 \rfloor, \lfloor 0.9(\frac{1}{t}\delta - 3\tau)m_s/m_0 \rfloor\}} \mathsf{Ext}_0(x, r_i)$$

where $r = \bigcirc_{i=1}^{\lfloor |r|/d_0 \rfloor} r_i \circ r'$ for some extra bits r' and $\forall i, |r_i| = d_0$.

Theorem 6.15. *For any constant $a \in \mathbb{N}^+$, any constant $c \in \mathbb{N}$, any $k = \delta n = \Theta(n/\log^c n)$, any $\epsilon = 1/2^{\Theta(\log^a n)}$ and any constant $\gamma \in (0,1)$, there exists an explicit construction of a strong (k, ϵ)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ in AC^0 of depth $\Theta(a+c)$, where $d = (\log n)^{\Theta(a+c)}$, $m = (1-\gamma)k$.*

Proof Sketch. The proof is almost the same as the proof of Theorem 6.13. We need to make sure that the parameters are correct.

We first claim that Ext in Construction 6.14 is a (k, ϵ)-extractor. The proof strategy is the same as that of Lemma 6.8 except that the parameters need to be modified. According to the same arguments as in the proof of Lemma 6.8, the overall error is

$$\tilde{\epsilon} \leq t(\epsilon_s + \epsilon_1) + tk(\epsilon_0 + 2^{-s}).$$

According to our settings, $\epsilon_s = O(\epsilon/n)$, $\epsilon_0 = \epsilon/(10tn)$, $2^{-s} \leq \epsilon/(10tn)$ and t is a constant. Also according to the definition of ϵ_1 in Lemma 6.8, ϵ_1 can be $1/2^{\Omega(k)}$. So $\tilde{\epsilon} \leq \epsilon$.

The seed length of Ext is $|U_d| = \sum_{i=1}^{t} |S_i| + |U_0| = (\log n)^{\Theta(a)}$.

The output length of Ext is $|Y_1| = \Theta(\delta k)$ which follows the same argument of that of Lemma 6.9.

According to our settings, Samp is in AC^0 of depth $\Theta(a)$ and Ext_1 is in AC^0 of depth $\Theta(a+c)$. Also we know that t is a constant. So Ext is in AC^0 of depth $\Theta(a+c)$.

The theorem holds according to the same argument (Extraction in parallel) as the proof of Theorem 6.13. The seed length increases to $(\log n)^{\Theta(a+c)}$ as we extract $\Theta(1/\delta)$ times in parallel.

□

7 Error Reduction For Sparse Extractors

Now we give error reduction methods for extractor families with small locality. As we do not require our construction to be in AC^0, the error can be exponentially small.

First we consider the construction of averaging samplers given by Vadhan [44] .

Lemma 7.1 ([44]). *For every $n \in \mathbb{N}$, $0 < \theta < \mu < 1$, $\gamma > 0$, there is a $(\mu, \mu-\theta, \gamma)$-averaging sampler $\mathsf{Samp} : \{0,1\}^r \to [n]^t$ that*

1. *outputs t distinct samples for $t \in [t_0, n]$, where $t_0 = O(\frac{\log(1/\gamma)}{\theta^2})$;*
2. *uses $r = \log(n/t) + \log(1/\gamma) \cdot \mathsf{poly}(1/\theta)$ random bits.*

Theorem 7.2. *For any constant $\delta \in (0,1]$, there exists an explicit construction of a $(k = \delta n, \epsilon)$-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ where ϵ can be as small as $2^{-\Omega(k)}$, $m = \Theta(\log(1/\epsilon))$, $d = O(\log(n/\epsilon))$ and the locality is $\Theta(\log(1/\epsilon))$.*

Proof. We follow a sample-then-extract procedure.

Let $\mathsf{Samp} : \{0,1\}^{r_s} \to \{0,1\}^t$ be a (μ_1, μ_2, γ)-averaging sampler following from Lemma 7.1. Let $\tau = 0.1\delta$, $\mu_1 = (\delta - 2\tau)/\log(1/\tau)$, $\mu_2 = (\delta - 3\tau)/\log(1/\tau)$.

For any (n,k)-source X, by Lemma 5.15, we have $\mathsf{SD}(R \circ X_{\mathsf{Samp}(R)}, R \circ W) \le \gamma + 2^{-\Omega(\tau n)}$. Here R is uniformly sampled from $\{0,1\}^{r_s}$. For every r in $\{0,1\}^{r_s}$, the random variable $W|_{R=r}$ is a $(t, (\delta-3\tau)t)$-source.

As δ is a constant, we know that τ is a constant. So $\theta = \mu_1 - \mu_2$ is a constant.

By Lemma 7.1, we can set $t = \Theta(\frac{\log(1/\gamma)}{\theta^2}) = \Theta(\log(1/\gamma))$, $r_s = \log(n/t) + \log(1/\gamma) \cdot \mathsf{poly}(1/\theta) = O(\log(\frac{n}{t\gamma}))$.

Let $\mathsf{Ext}_1 : \{0,1\}^t \times \{0,1\}^{d_1} \to \{0,1\}^m$ be a $((\delta-3\tau)t, \epsilon_1)$-extractor following from Lemma 2.7 where ϵ_1 can be $2^{-\Omega(k)}$. As a result,

$$\mathsf{SD}(U \circ \mathsf{Ext}_1(W, U), U') \le \epsilon_1,$$

where U, U' are uniform distributions. Also $d_1 = O(\log(t/\epsilon_1))$, $m = \Theta(\delta t) = \Theta(\log(1/\gamma))$.

As a result, the sample-then-extract procedure gives an extractor with the error

$$\gamma + 2^{-\Omega(\tau n)} + \epsilon_1.$$

According to our settings, $\gamma + 2^{-\Omega(\tau n)}$ can be as small as $2^{-\Omega(k)}$ when $\gamma = 2^{-\Omega(k)}$ and ϵ_1 can be $2^{-\Omega(k)}$.

Thus the error of the extractor can be $2^{-\Omega(k)}$ by setting $2^{-\Omega(\tau n)} \le 0.1\epsilon, \gamma = 0.1\epsilon, \epsilon_1 = 0.1\epsilon$.

The seed length is $r_s + d_1 = \Theta(\log(n/t) + \log(1/\gamma)) + \Theta(\log(t/\epsilon_1)) = \Theta(\log(n/\epsilon))$.

The locality is $t = O(\log(1/\epsilon))$. This is because when the seed is fixed, we select t bits from X by sampling. After that, we apply Ext_1 on these t bits. So each output bit depends on t bits when the seed is fixed.

The output length $m = \Theta(\delta t) = \Theta(\log(1/\gamma)) = \Theta(\log(1/\epsilon))$. □

Next we construct extractors with small locality and exponentially small errors.

First we give the construction for error reduction.

Construction 7.3 (Error Reduction for Sparse Extractors with Exponentially Small Errors). *For any $k = \frac{n}{\mathsf{poly}(\log n)}$, let X be an (n,k)-source. We construct a strong (k,ϵ)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ where ϵ can be as small as $2^{-k^{\Omega(1)}}$, $d = \Theta(\log n + \frac{\log^2(1/\epsilon)}{\log n})$, $m = k^{\Theta(1)}$.*

- *Let $\mathsf{Ext}_0 : \{0,1\}^{n_0=n} \times \{0,1\}^{d_0} \to \{0,1\}^{m_0}$ be a (k_0, ϵ_0)-extractor following from Theorem 4.11 where $k_0 = k - \Delta_1, \Delta_1 = \log(n/\epsilon)$, $\epsilon_0 = k^{-\Theta(1)}$, $d_0 = \Theta(\log n)$, $m_0 = k^{\Theta(1)}$.*

- *Let* $\mathsf{Ext}_1 : \{0,1\}^{n_1=m_0/t_2} \times \{0,1\}^{d_1} \to \{0,1\}^{m_1}$ *be a* (k_1, ϵ_1)*-extractor following from Theorem 7.2 where* $k_1 = 0.9n_1$, $\epsilon_1 = 2^{-\Omega(k_1)}$, $d_1 = O(\log(n/\epsilon_1))$, $m_1 = \Theta(\log(1/\epsilon_1))$.
- *Let* t_1 *be such that* $(2\epsilon_0)^{t_1} \leq 0.1\epsilon$. *(We only consider the case that* $\epsilon < \epsilon_0$. *If* $\epsilon \geq \epsilon_0$, *we set* Ext *to be* Ext_0.*)*
- *Let* $t_2 = m_0^{1/3}$.

Our construction is as follows.

1. *Let* $R_1, R_2, \ldots, R_{t_1}$ *be independent uniform distributions such that for every* $i \in [t_1]$ *the length of* R_i *is* d_0. *Get* $Y_1 = \mathsf{Ext}_0(X, R_1), \ldots, Y_{t_1} = \mathsf{Ext}_0(X, R_{t_1})$.
2. *Get* $Y = Y_1 \circ Y_2 \circ Y_3 \circ \cdots \circ Y_{t_1}$.
3. *For each* $i \in [t_1]$, *let* $Y_i = Y_{i,1} \circ Y_{i,2} \circ \cdots \circ Y_{i,t_2}$ *such that for every* $j \in [t_2]$, $Y_{i,j}$ *has length* $n_1 = m_0/t_2$. *Let* $S_1, S_2, \ldots, S_{t_1}$ *be independent uniform distributions, each having length* d_1. *Get* $Z_{i,j} = \mathsf{Ext}_1(Y_{i,j}, S_i), \forall i \in [t_1], j \in [t_2]$. *Let* $Z_i = Z_{i,1} \circ Z_{i,2} \circ \cdots Z_{i,t_2}$.
4. *Let* $R = \bigcirc_i R_i, S = \bigcirc_i S_i$. *We get* $\mathsf{Ext}(X, U) = Z = \bigoplus_i^{t_1} Z_i$ *where* $U = R \circ S$.

Theorem 7.4. *For any* $k = \frac{n}{\mathsf{poly}(\log n)}$, *there exists an explicit construction of a strong* (k, ϵ)*-extractor* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, *where* ϵ *can be as small as* $2^{-k^{\Omega(1)}}$, $d = \Theta(\log n + \frac{\log^2(1/\epsilon)}{\log n})$, $m = \Theta(\delta n \frac{1}{10800} \log(\frac{1}{\epsilon})) = k^{\Theta(1)}$ *and the locality is* $\log^2(1/\epsilon)\mathsf{poly}(\log n)$.

Proof Sketch. The proof is almost the same as that of Theorem 5.13. We only need to make sure the parameters are correct.

There are two differences between Construction 7.3 and Construction 5.6. First, in Construction 7.3, the extractor Ext_1 follows from Theorem 7.1. Second, Construction 7.3 works for ϵ which can be as small as $2^{-k^{\Omega(1)}}$.

We first claim that the function Ext is a (k, ϵ)-extractor. The proof strategy is the same as that of Lemma 5.7, except that the parameters need to be modified.

As $(2\epsilon_0)^{t_1} \leq 0.1\epsilon$, $t_1 = O(\frac{\log(1/\epsilon)}{\log n})$.

According to the proof of Lemma 5.7, the error for Ext is

$$(2\epsilon_0)^{t_1} + (2^{-\Delta_1} + \epsilon' + (\epsilon_0' + \epsilon_1)t_2)$$

Here ϵ', ϵ_0' follows the same definitions as that in the proof of Lemma 5.7. As a result, we know that both ϵ' and ϵ_0' can be $2^{-k^{\Omega(1)}}$. Also $\Delta_1 = \log(n/\epsilon)$, $\epsilon_1 = \epsilon/n$.

As a result $\epsilon = 2^{-\Delta_1} + \epsilon' + (\epsilon_0' + \epsilon_1)t_2$ can be $2^{-k^{\Omega(1)}}$ and the overall error can be at least $2^{-k^{\Omega(1)}}$.

By the same proof as that of Lemma 5.10, we know that the output length is $m = \Theta(n^{\frac{1}{10800}}\delta \log(\frac{1}{\epsilon}))$.

For the seed length, we know that according to the settings of Ext_0 and Ext_1, as $t_1 = O(\frac{\log(1/\epsilon)}{\log n})$, if $t_1 = \omega(1)$ then $d = \Theta(t_1 d_0 + t_1 d_1) = \Theta(\frac{\log^2(1/\epsilon)}{\log n})$. If $t = O(1)$, the seed length should be $\Theta(\log n)$ as we need at least run Ext_0 for once. As a result, the seed length is $\Theta(\log n + \frac{\log^2(1/\epsilon)}{\log n})$.

As the locality of Ext_0 is $l_0 = \mathsf{poly}(\log n)$ and the locality of Ext_1 is $l_1 = O(\log(1/\epsilon))$, the locality of Ext is $t_1 \times l_1 \times l_0 = \log^2(1/\epsilon)\mathsf{poly}(\log n)$.

□

Theorem 7.5. *For any $k = \delta n = \frac{n}{\mathsf{poly}(\log n)}$ and any constant $\gamma \in (0,1)$, there exists an explicit construction of a strong (k,ϵ)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, where ϵ can be as small as $2^{-k^{\Omega(1)}}$, $d = k^\alpha$ for some constant $\alpha \in (0,1]$, $m = (1-\gamma)k$ and the locality is $\log^2(1/\epsilon)\mathsf{poly}(\log n)$.*

Proof. Let $\epsilon = 2^{-k^\beta}$ for some very small constant $\beta < 1$. Let $\mathsf{Ext}_0 : \{0,1\}^{n_0} \times \{0,1\}^{d_0} \to \{0,1\}^{m_0}$ be a $(k_0, \epsilon_0 = \epsilon/n)$-extractor following from Theorem 7.4. Here $k_0 = k - tm_0 - s$ where $s = \log(n/\epsilon)$, $t = (1-\gamma)k/m_0$. By lemma 6.4, we know that there exists a (k,ϵ')-extractor Ext with $\epsilon' = t(\epsilon_0 + 2^{-s}) \le \epsilon$. The output length is $(1-\gamma)k$.

According to the construction in Lemma 6.4, the locality of Ext is the same as that of Ext_0 which is $\log^2(1/\epsilon)\mathsf{poly}(\log n)$. The seed length is $t \times \Theta(\log n + \frac{\log^2(1/\epsilon)}{\log n}) = k^\alpha$ for some $\alpha \in (0,1]$ as β can be small enough. □

8 Deterministic Extractor for Bit-fixing Source

We use two crucial tools. One is the extractor for non-oblivious bit-fixing sources, proposed by Chattopadhyay and Zuckerman [12] and improved by Li [32].

Theorem 8.1 ([32] Theorem 1.11)**.** *Let c be a constant. For any $\beta > 0$ and all $n \in \mathbb{N}$, there exists an explicit extractor $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^m$ such that for any (q,t,γ)-non-oblivious bit-fixing source X on n bits with $q \le n^{1-\beta}$, $t \ge c\log^{21} n$ and $\gamma \le 1/n^{t+1}$,*

$$\mathsf{SD}(\mathsf{Ext}(X), U) \le \epsilon$$

where $m = t^{\Omega(1)}$, $\epsilon = n^{-\Omega(1)}$.

The extractor can be computed by standard circuits of depth $\lceil \frac{\log m}{\log\log n} \rceil + O(1)$.

To see the depth is $\lceil \frac{\log m}{\log\log n} \rceil + O(1)$, let's briefly recall the construction of the extractor. It first divides the input in to $n^{O(1)}$ blocks. Then for each block, it applys the extractor from [32] Theorem 4.1 which has depth 4. At last, it conducts a multiplication between a matrix of size $m \times O(m)$ and a vector of dimension $O(m)$, both over $\mathbb{F}_2$. The last step can be computed by a circuit of depth $\lceil \frac{\log m}{\log\log n} \rceil + 1$ by Lemma 2.11 part 2.

The other tool is the design extractor introduced by Li [31].

Definition 8.2 ([31])**.** *An $(N, M, K, D, \alpha, \epsilon)$-design extractor is a bipartite graph with left vertex set $[N]$, right vertex set $[M]$, left degree D such that the following properties hold.*

- *(extractor property) For any subset $S \subseteq [M]$, let $\rho_S = |S|/M$. For any vertex $v \in [N]$, let $\rho_v = |\Gamma(v) \cap S|/D$. Let $Bad_S = \{v \in [N] : |\rho_v - \rho_S| > \epsilon\}$, then $|Bad_S| \le K$. ($\Gamma(\cdot)$ outputs the set of all neighbors of the input.)*
- *(design property) For any two different vertices $u, v \in [N]$, $|\Gamma(u) \cap \Gamma(v)| \le \alpha D$.*

Construction 8.3. *For any constant $a \in \mathbb{N}$, any $t = \mathsf{poly}(\log n)$, the deterministic extractor $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^{m = t^{\Omega(1)}}$ for any $(n, \delta n = \Theta(n/\log^a n))$-bit-fixing source is constructed as the follows.*

- *Construct an $(N, M, K, D, \alpha, \epsilon)$-design extractor, where $M = n, K = n^{1/0.9}, N = n^{1/0.3}$, $\epsilon = 1/\log^c N$, $D = \log^b N$, $\alpha = D/M + \epsilon$, for $c = \lceil \log t/\log\log N \rceil + a + 1$ and large enough constant $b = \Theta(c)$.*
- *Let $Y = (Y_1, Y_2, \ldots, Y_N)$. Compute $Y_i = \bigoplus_{j \in \Gamma(i)} X_j$, for $i = 1, \ldots, N$, by taking i as the ith vertex in the left set of the design extractor.*

- *Let* $\mathsf{Ext}(X) = \mathsf{Ext}'(Y)$ *where* $\mathsf{Ext}' : \{0,1\}^N \to \{0,1\}^m$ *is the extractor from Theorem 8.1 with error* $\epsilon = n^{-\Omega(1)}$.

Lemma 8.4. *An* $(N, M, K, D, \alpha, \epsilon)$*-design extractor, where* $K = N^{1/3}, M = K^{0.9}$, $\epsilon = 1/\log^c N$, $D = \log^b N$,$\alpha = D/M + \epsilon$, *for any constant* c *and large enough constant* $b = \Theta(c)$, *can be constructed in polynomial time.*

Proof. In [31] it is showed that design extractors can be constructed in deterministic polynomial time by a greedy algorithm.

The construction is based on a (k_0, ϵ)-extractor $\mathsf{Ext}_0 : \{0,1\}^{n_0} \times \{0,1\}^{d_0} \to \{0,1\}^{m_0}$, from Theorem 2.7 (almost optimal parameters), for any (n_0, k_0)-source, where $n_0 = 4k_0, m_0 = 0.9k_0, d_0 = O(\log(n_0/\epsilon))$. Also we substitute the first d_0 bits of the output by the seed s.t. every left vertex has exactly 2^{d_0} neighbors. Recall the greedy algorithm proposed by Li [31], which picks vertices one by one, deleting the vertices which does not meet the design property before each picking. At last we can get $2^{n_0-k_0} \geq 2^{3k_0}$ left vertices. Let $N = 2^{3k_0}, K = 2^{k_0}, M = 2^{m_0}$. We know that $\epsilon = 1/\log^c N$ and $d_0 = O(\log(n_0/\epsilon))$. Thus if $b = \Theta(c)$ is large enough, D can be $\log^b N$ by adding extra random bits to adjust the length of the seed.

□

Lemma 8.5. *For any constant* $a \in \mathbb{N}$, *any* $t = \mathsf{poly}(\log n)$, *if* X *is an* $(M, \delta M = \Theta(M/\log^a M))$*-bit-fixing source, then* $Y = g(X)$ *is a* $(q, t, 0)$*-non-oblivious bit-fixing source, where* $q = K$.

Proof. Assume the coordinates of random bits of X form the set S. By the extractor property of design extractors, the number of left vertex x, such that $|\rho_x - \rho_S| > \epsilon$, is at most K. These vertices form the set Bad_S.

We prove that for any subset $V \subseteq [N] \backslash Bad_S$ with size $|V| \leq t$, $\bigoplus_{j \in V} Y_j$ is uniformly distributed.

Let $V = \{v_1, v_2, \ldots, v_{t'}\}$ be a subset of $[N] \backslash Bad_S$, where $t' \leq t$. So $|\Gamma(v_{t'}) \cap S| \geq (\delta - \epsilon)D$. By the design property of design extractors, for any $i = 1, 2, \ldots, t' - 1, |\Gamma(v_{t'}) \cap \Gamma(v_i)| \leq \alpha D$. So $|(\Gamma(v_{t'}) \cap S) \backslash \bigcup_{i=1}^{t'-1} \Gamma(v_i)| \geq (\delta - \epsilon)D - t \cdot \alpha D \geq 1$ for $c = \lceil \log t / \log \log N \rceil + a + 1$ and large enough constant b. Thus $\bigoplus_{j \in V} Y_j$ is uniformly distributed because some uniform random bits in $\Gamma(v_{t'}) \cap S$ cannot be canceled out by bits in $\bigcup_{i=1}^{t'-1} \Gamma(v_i)$.

By the Information Theoretic XOR-Lemma in [17], $Y_{[N] \backslash Bad_S}$ is t-wise independent. Thus $Y = g(X)$ is a $(q = K, t, 0)$-non-oblivious bit-fixing source.

□

Theorem 8.6. *For any constant* $a \in \mathbb{N}$, *there exists an explicit deterministic* $(k = \delta n = \Theta(n/\log^a n), \epsilon = n^{-\Omega(1)})$*-extractor* $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^m$ *that can be computed by* AC^0 *circuits of depth* $\Theta(\frac{\log m}{\log \log n} + a)$, *for any* (n, k)*-bit-fixing source, where* m *can be any* $\mathsf{poly}(\log n)$.

Proof. We claim that Construction 8.3 gives the desired extractor. Let X be an (n, k)-bit-fixing source. By lemma 8.5, we get Y which is a $(q, t, 0)$-non-oblivious bit-fixing source with length $\Theta(n^{1/0.3})$, where $q = \Theta(n^{1/0.9})$ and t can be any large enough $\mathsf{poly}(\log n)$.

By Theorem 8.1, $\mathsf{Ext}(X) = \mathsf{Ext}'(Y)$ is ϵ-close to uniform. The output length $m = t^{\Omega(1)}$ can be any $\mathsf{poly}(\log n)$ as we can set t to be any $\mathsf{poly}(\log n)$.

We show that the circuit for computing the extractor is in uniform AC^0. In Construction 8.3, each Y_i is the XOR of poly-logarithmic bits of X. Also the extractor of Theorem 8.1 is in AC^0. So the overall construction is in AC^0. It is in uniform AC^0 because the design extractor can be constructed in polynomial time by Lemma 8.4 while all other operations are explicit and can be computed by uniform AC^0 circuits.

The depth of the circuit is $\Theta(\frac{\log m}{\log\log n} + a)$. Because in Construction 8.3, $c = \lceil \log t / \log\log N \rceil + a + 1$ and $b = \Theta(c)$. By lemma 2.11 part 2, the XOR of $D = 1/\log^b N$ bits can be computed by circuits of depth b. Also the depth of Ext' is $\lceil \frac{\log m}{\log\log n} \rceil + O(1)$. Thus the overall depth is $\Theta(\frac{\log m}{\log\log n} + a)$.

□

Next we do error reduction. Our method is based on the XOR lemma given by Barak, Impagliazzo and Wigderson [7].

Lemma 8.7 ([7] Lemma 3.15)**.** *Let $Y_1, Y_2, \ldots, Y_t$ be independent distributions over $\mathbb{F}$ such that $\forall i \in [t], \mathsf{SD}(Y_i, U) \leq \epsilon$. Then*

$$\mathsf{SD}(\sum_{i=1}^{t} Y_i, U) \leq (2\epsilon)^t,$$

where U is uniform over $\mathbb{F}$.

Proof. For simplicity, let $\mathbb{F} = \{0, 1, \ldots, M-1\}$.

We use induction to show that for $j = 1, 2, \ldots, t$, $\mathsf{SD}(\sum_{i=1}^{j} Y_i, U) \leq (2\epsilon)^j$.

As Y_1 is ϵ-close to uniform, this shows the base case.

Let $Y' = \sum_{i=1}^{j-1} Y_i$. Suppose $\mathsf{SD}(Y', U) \leq (2\epsilon)^{j-1}$.

Let $p' = (p'_0, p'_1, \ldots, p'_{M-1})$ be such that $p'_i = \Pr[Y' = i_b] = 1/M + \delta'_i$, for $i = 0, 1, \ldots, M-1$, where i_b is the binary form of i.

We know that $\mathsf{SD}(Y', U) = 1/2(\sum_{i=1}^{M-1} |\delta'_i|)$ and $\sum_{i=0}^{M-1} \delta'_i = 0$.

Let $p = (p_0, p_1, \ldots, p_{M-1})$ be such that $p_i = \Pr[Y_j = i_b] = 1/M + \delta_i$ for $i = 0, 1, \ldots, M-1$. We know that $\mathsf{SD}(Y_j, U) = 1/2(\sum_{i=1}^{M-1} |\delta_i|)$ and $\sum_{i=0}^{M-1} \delta_i = 0$.

So

$$\begin{aligned}
\Pr[Y' + Y_j = i_b]| &= \sum_{k=0}^{M-1} \Pr[Y' = k_b] \Pr[Y_j = (i-k)_b] \\
&= \sum_{k=0}^{M-1} p'_k \cdot p_{i-k} \\
&= 1/M + 2(\sum_{k=0}^{M-1} \delta_k)/M + \sum_{k=0}^{M-1} \delta_k \delta_{i-k} \\
&= 1/M + \sum_{k=0}^{M-1} \delta_k \delta_{i-k}.
\end{aligned} \tag{21}$$

Thus $|\Pr[Y' + Y_j = i_b] - \Pr[U = i_b]| = |\sum_{k=0}^{M-1} \delta_k \delta_{i-k}|$.

As a result,

$$\begin{aligned}
\mathsf{SD}(Y' + Y_j, U) &= 1/2 \sum_{i=0}^{M-1} |\Pr[Y' \oplus Y_j = i_b] - \Pr[U = i_b]| \\
&= 1/2 \sum_{i=0}^{M-1} |\sum_{k=1}^{M-1} \delta_k \delta_{i-k}| \\
&\leq 1/2 (\sum_{k=0}^{M-1} \sum_{l=0}^{M-1} |\delta_k \delta_l|) \\
&= 1/2 (\sum_{i=1}^{M-1} |\delta'_i|)(\sum_{i=1}^{M-1} |\delta_i|) \\
&\leq (2\epsilon)^j.
\end{aligned} \tag{22}$$

□

Theorem 8.8. *If* $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^m$ *is a* (k, ϵ)*-extractor for* (n, k)*-bit-fixing sources, then for every* $l \in \mathbb{N}$*, the function* $\mathsf{Ext}' : \{0,1\}^{ln} \to \{0,1\}^m$*, given by* $\mathsf{Ext}'(x_1, \ldots, x_l) = \oplus_{i \in [l]} \mathsf{Ext}(x_i)$ *is a* $(2lk, \epsilon^{\Theta(l)})$*-extractor for* $(ln, 2lk)$*-bit-fixing sources.*

Proof. Let $X = (X^{(1)}, \ldots, X^{(l)})$ be an $(ln, 2lk)$-bit-fixing source. Then for $\delta = k/n$ fraction of $j \in [l]$, $X^{(j)}$ is an $(n, \delta n)$-bit-fixing source. Because if not, the total number of random bits is at most $\delta l \cdot n + (1 - \delta) l \delta n < 2\delta l n = 2lk$. Also we know that $X^{(1)}, X^{(2)}, \ldots, X^{(l)}$ are independent because X is a bit-fixing source. We regard each $\mathsf{Ext}(X^{(i)})$ as a random element (coefficients of the corresponding polynomial) in $\mathbb{F}_{2^m}$. By Lemma 8.7, $\mathsf{Ext}'(X)$ is $\epsilon^{\Theta(l)}$-close to uniform.

□

By using the deterministic extractor in Theorem 8.6, combining with Theorem 8.8, adjusting the parameters, we get the following result.

Theorem 8.9. *For any constant* $a, c \in \mathbb{N}$ *, there exists an explicit deterministic* $(k = \Theta(n/\log^a n), \epsilon = 2^{-\log^c n})$*-extractor* $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^m$ *that can be computed by* AC^0 *circuits of depth* $\Theta(\frac{\log m}{\log \log n} + a + c)$*, for any* (n, k)*-bit-fixing sources, where* m *can be any* $\mathsf{poly} \log n$*.*

Finally we do output length optimization by applying the same technique as that in [19]. The technique is given by Gabizon et al.[14].

Theorem 8.10. *For any constant* $a, c \in \mathbb{N}$ *and any constant* $\gamma \in (0, 1]$*, there exists an explicit deterministic* $(k = \Theta(n/\log^a n), \epsilon = 2^{-\log^c n})$*-extractor* $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^{(1-\gamma)k}$ *that can be computed by* AC^0 *circuits of depth* $\Theta(a + c)$*, for any* (n, k)*-bit-fixing sources.*

proof sketch. The difference between our construction and [19] Theorem 5.12 is that, for the three crucial components in the construction, we use the deterministic extractor of Theorem 8.9, the seeded extractor of Theorem 6.15 and the averaging sampler in Lemma 5.18 instead. We briefly describe the construction as the follows.

- A deterministic ϵ_1-error extractor $\mathsf{Ext}_1 : \{0,1\}^n \to \{0,1\}^{r+r_2}$ for $(n, \mu' s)$-bit-fixing sources, by Theorem 8.9;
- A seeded ϵ_2-error extractor $\mathsf{Ext}_2 : \{0,1\}^n \times \{0,1\}^{r_2} \to \{0,1\}^m$ for $(n, \mu n - s)$-bit-fixing sources, by Theorem 6.15;

- An (μ, μ', θ)-averaging sampler $\mathsf{Samp} : \{0,1\}^r \to [n]^s$, by Lemma 5.18 .

We set $\mu = k/n, \mu' = \mu/2, s = k/2$ such that $\mu' s = \Theta(n/\log^{2a} n)$ and $\mu n - s = \Theta(n/\log^a n)$. Also we set $\epsilon_1 = 2^{-\log^{3c} n}$, $\epsilon_2 = \theta = 2^{-\log^{2c} n}$, $m = (1-\gamma)k$, $r_2 = (\log n)^{\Theta(a+c)}$ and $r = \Theta(\log^{2c} n)$.

Theorem 7.1 of [14] constructs a deterministic extractor $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^m$ for $(n, \mu n)$-bit-fixing sources, where the error is $\epsilon = \epsilon_2 + 2^{r+3}\epsilon_1 + 3\theta$. So $\epsilon \leq 2^{-\log^c n}$. The construction is $\mathsf{Ext}(x) = \mathsf{Ext}_2(x_{[n]\setminus S(Z_1)} \circ 0^t, Z_2)$, where Z_1 is the first r bits of $\mathsf{Ext}_1(X)$ and Z_2 is the last r_2 bits of $\mathsf{Ext}_1(X)$.

Here we only need to compute the depth of the circuit. We know that $r + r_2 = (\log n)^{\Theta(a+c)}$. The depth of the final extractor is the sum of the depths of all three components. By Theorem 8.9, the depth for the deterministic extractor is $\Theta(a+c)$. By theorem 6.15, the depth for the seeded extractor is also $\Theta(a+c)$. By Lemma 5.18, the depth for the sampler is $\Theta(c)$. So the overall depth is $\Theta(a+c)$.

□

9 Randomness Condenser with Small Locality

In this section, we consider constructions of extractor families with small locality by first constructing a condenser family with small locality. Our condenser will be based on random walks on expander graphs and pseudorandom generators for space bounded computation.

9.1 Basic Construction

We give a condense-then-extract procedure by first constructing a randomness condenser.

Theorem 9.1 (Hitting Property of Random Walks [29] Theorem 23.6)**.** *Let $G = (V, E)$ be a d-regular graph with $\lambda(G) = \lambda$. Let $B \subseteq V$ such that $|B| = \beta|V|$. Let (B, t) be the event that a random walk of length t stays in B. Then $\Pr[(B,t)] \leq (\beta + \lambda)^t$.*

Lemma 9.2 ([5] implicit)**.** *For every $n \in \mathbb{N}$ and for every $0 < \alpha < 1$, there is an explicit construction of d-regular Graphs G_n which have the following properties.*

1. *$\lambda \leq \alpha$.*
2. *G_n has n vertices.*
3. *d is a constant.*
4. *There exists a $\mathsf{poly}(\log n)$-time algorithm that given a label of a vertex v in G_n and an index $i \in d$, output the ith neighbor of v in G_n.*

Lemma 9.3 ([41])**.** *Let $H_2(X) = \log(1/\mathsf{coll}(X))$, $\mathsf{coll}(X) = \Pr_{X_1,X_2}[X_1 = X_2]$, where X_1, X_2 are independent random variables having the same distribution as X.*

For any random variable X,

$$2H_\infty(X) \geq H_2(X).$$

Recall that $w(\cdot)$ denotes the weight of the input string as we defined in Section 2.

Lemma 9.4. *Given any (n,k)-source X and any string $x \in \{0,1\}^n$, with probability $1-2^{-0.5k}$, $w(X \oplus x) \geq k/(c_1 \log n)$ for any constant $c_1 \geq 2$; with probability $1 - 2^{-0.5k}$, $w(X \oplus x) \leq n - k/(c_1 \log n)$ for any constant $c_1 \geq 2$.*

Proof. The number of strings which have i digits different from x is $\binom{n}{i}$. So the number of strings which have at most $l = k/(c_1 \log n)$ digits different from x is at most $\sum_{i=0}^{l} \binom{n}{i} \leq (\frac{en}{l})^l \leq 2^{0.5k}$ for any constant $c_1 \geq 2$. So with probability at least $1 - 2^{-0.5k}$, $w(X \oplus x) \geq l$.

Also as $\sum_{i=n-l}^{n} \binom{n}{i} = \sum_{i=0}^{l} \binom{n}{i}$, with probability $1 - 2^{-0.5k}$, $w(X \oplus x) \leq n - l$. □

Lemma 9.5. *Consider a random vector $v \in \{0,1\}^n$ where $v_1, \ldots, v_n$ are independent random bits and $\forall i, \Pr[v_i = 1] = p = 1/\mathsf{poly}(n)$. For any $\epsilon = 1/\mathsf{poly}(n)$, there is an explicit function $f : \{0,1\}^l \to \{0,1\}^n$ where $l = O(n \log n)$, such that*

$$\forall i \in [n], |\Pr[f(U)_i = 1] - p| \leq \epsilon$$

where U is the uniform distribution of length l.

There exists an algorithm A which runs in $O(\log n)$ space and can compute $f(s)_i$, given input $s \in \{0,1\}^l$ and $i \in [n]$.

Proof. We give the algorithm A which runs in $O(\log n)$ space and can compute $f(s)_i$, given input $s \in \{0,1\}^l$ and $i \in [n]$.

Assume the binary expression of p is $0.b_1 b_2 b_3 \ldots$. The algorithm A is as follows. Intuitively, A divides s into n blocks and uses the ith block to generate a bit which simulates v_i roughly according to the probability p.

1. Assume that s has n blocks. The ith block is s_i where $|s_i| = t = c \log n$ for some constant c. Let $j = 1$.
2. If $j = t + 1$, go to step 3. If $s_{i,j} < b_j$, then set $f(s)_i = 1$ and stop; if $s_{i,j} > b_j$, set $f(s)_i = 0$ and stop; if $s_{i,j} = b_j$, set $j = j + 1$ and go to step 2.
3. Set $f(s)_i = 1$ and stop.

For every i, the probability

$$\Pr[f(s)_i = 1] = \Pr[0.s_{i,1} \ldots s_{i,t} \leq 0.b_1 b_2 \ldots b_t] = 0.b_1 b_2 \ldots b_t.$$

As a result,

$$\forall i \in [n], |\Pr[f(s)_i = 1] - \Pr[v_i = 1]| \leq 0.00 \ldots b_{t+1} b_{t+2} \ldots \leq 2^{-t} = 2^{-c \log n}.$$

For any $\epsilon = 1/\mathsf{poly}(n)$, if c is large enough, then $2^{-t} = 2^{-c \log n} \leq \epsilon$.

The input length for f is $l = n \times t = O(n \log n)$.

As all the iterators and variables in A only need $O(\log n)$ space, A runs in space $O(\log n)$. This proves our conclusion.

□

Construction 9.6. *For any $k = \Omega(\log^2 n)$, we construct an $(n, k, t = 10k, 0.1k, \epsilon_c = 2^{-0.1k})$-condenser $\mathsf{Cond} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^t$ with $d = O(n \log n)$ and locality $c = n/l, l = \frac{k}{2 \log n}$.*

1. *Construct an expander graph $G = (V, E)$ where $V = \{0,1\}^{r_0 = O(n \log n)}$ and $\lambda = 0.01$.*
2. *Use a uniform random string U_1 of length r_0 to select a vertex v_1 of V.*
3. *Take a random walk on G starting from v_1 to get $v_2, \ldots, v_t$ for $t = 10k$.*

4. *For $i \in [t]$, get an n-bit string $v'_i = f(v_i)$ such that $\forall j \in [n], \Pr[v'_{i,j} = 1] - 1/l| \leq 1/n^2$, where $f : \{0,1\}^{r_0} \to \{0,1\}^n$ follows from Lemma 9.5, $r_0 = O(n \log n)$.*

5. *Let $M = (v'_1, \ldots, v'_t)^T$.*

6. *Let $\mathsf{Cond}(x, u) = Mx$.*

Let $V' = \{0,1\}^n$. Let $B_i = \{v \in V' : w(v) = i\}, i \in [d]$. Let $A_{x,j} = \{v \in B_j : \langle v, x\rangle = 0\}$.
For any $x \in \{0,1\}^n$, let $V_x = \{v \in V : \langle f(v), x\rangle = 0\}$. Let $T = \{v \in V : w(f(v)) \in [0.8c, 1.2c]\}$.

Lemma 9.7. *In Construction 9.6, with probability $1 - 2\exp\{-\Theta(c)\}$, $w(v'_1) \in [0.8c, 1.2c]$. That is,*

$$\frac{|T|}{|V|} \geq 1 - 2\exp\{-\Theta(c)\}.$$

Proof. According to our construction, $\forall i \in [n], \Pr[v'_{1,i} = 1] \in [1/l - 1/n^2, 1/l + 1/n^2]$. As a result, $\mathbf{E}w(v'_1) \in [n/l - 1/n, n/l + 1/n] = [c - 1/n, c + 1/n]$. According to Chernoff Bound, we know that $\Pr[w(v'_1) \in [0.8c, 1.2c]] \geq \Pr[w(v'_1) \in [0.9\mathbf{E}w(v'_1), 1.1\mathbf{E}w(v'_1)]] \geq 1 - 2\exp\{-\Theta(c)\}$.

□

Lemma 9.8. *In construction 9.6, we have the following conclusions.*

1. *For any $x \in \{0,1\}^n$ with $w(x) \in [l, n-l]$, for any integer $j \in [0.8c, 1.2c]$, $\beta_{x,j} = |A_{x,j}|/|B_j| \leq 5/6$.*

2. *For any $x \in \{0,1\}^n$ with $w(x) \in [l, n-l]$, $|V_x|/|V| \leq 5/6 + 1/\mathsf{poly}(n)$.*

3. *$\Pr[MX_1 = MX_2] = p_0 \leq 2^{-0.5k+1} + (5/6 + 1/\mathsf{poly}(n) + \lambda)^t$ where X_1, X_2 are independent random variables that both have the same distribution as X.*

4. *$U_d \circ \mathsf{Cond}(X, U_d)$ is $p_0^{1/3}$-close to $U_d \circ W$. Here U_d is a uniform distribution of length d. For every u, $W|_{U_d=u}$ has entropy $\frac{1}{3}\log\frac{1}{p_0}$.*

5. *$U_d \circ \mathsf{Cond}(X, U_d)$ is $\epsilon_c = 2^{-0.1k}$-close to $U_d \circ W$ where $\forall u \in \{0,1\}^d$, $W|_{U_d=u}$ has entropy $d + 0.1k$.*

Proof. We know that

$$\beta_{x,j} = \frac{1}{2}(1 + \frac{\sum_{i=0}^{\min\{w(x),j\}} \binom{w(x)}{i}\binom{n-w(x)}{j-i}(-1)^i}{\binom{n}{j}}).$$

Because $\langle v, x\rangle = 0$ happens if and only if $|\{i \in [n] : x_i = v_i = 1\}|$ is even.
Let $\Delta = \sum_{i=0}^{\min\{w(x),j\}} \binom{w(x)}{i}\binom{n-w(x)}{j-i}(-1)^i$.
Consider the series $\Delta_i = \binom{w(x)}{i}\binom{n-w(x)}{j-i}, i = 0, 1, \ldots, \min\{w(x), j\}$. Let $\Delta_i \geq \Delta_{i+1}$, we can get

$$i \geq \frac{jw(x) - n + w(x) + j - 1}{n+2} \in [\frac{0.8w(x)}{l} - 3, \frac{1.2w(x)}{l}]$$

Let $\Delta_{i-1} \leq \Delta_i$, we can get

$$i \leq \frac{jw(x) + w(x) + j + 1}{n-2} \in [\frac{0.8w(x)}{l}, \frac{1.2w(x)}{l} + 3].$$

So series $\Delta_i, i = 0, 1, \ldots, \min\{w(x), j\}$ has its maximum for some $i \in [\frac{0.8w(x)}{l} - 3, \frac{1.2w(x)}{l} + 3]$.
To make it simpler, first we consider the situation that $0 < \frac{w(x)}{l} - 3$ and $j > \frac{w(x)}{l} + 3$.

Let $i' = \arg\max_i\{\Delta_i\}$ which is in $[\frac{w(x)}{l} - 3, \frac{w(x)}{l} + 3]$. Consider $\Delta_{i'}/\Delta_{i'-1}$. Let $i' = \frac{\theta w(x)}{l} + \delta$ for some $\theta \in [0.8, 1.2], \delta \in [-3, 3]$.

$$\begin{aligned}\frac{\Delta_{i'}}{\Delta_{i'-1}} &= \frac{w(x) - i' + 1}{i'} \cdot \frac{j - i' + 1}{n - w(x) - j + i'} \\ &= \frac{w(x) - \frac{\theta}{l}w(x) - \delta + 1}{\frac{\theta}{l}w(x) + \delta} \cdot \frac{j - \frac{\theta}{l}w(x) - \delta + 1}{n - w(x) - j + \frac{\theta}{l}w(x) + \delta}\end{aligned} \tag{23}$$

The first term

$$\frac{w(x) - \frac{\theta}{l}w(x) - \delta + 1}{\frac{\theta}{l}w(x) + \delta} = l + O(1).$$

As $j \in [0.8c, 1.2c]$, we know

$$\frac{n - w(x) - j + \frac{\theta}{l}w(x) + \delta}{j - \frac{\theta}{l}w(x) - \delta + 1} \geq \frac{n - w(x) - 0.8c + \frac{\theta}{l}w(x) + 3}{0.8c - \frac{\theta}{l}w(x) - 2} \geq \frac{5l}{6}$$

So $\frac{\Delta_{i'}}{\Delta_{i'-1}} \leq 2$.

As $\binom{n}{c} \geq \Delta_{i'} + \Delta_{i'-1}$, we know $\frac{\Delta_{i'}}{\binom{n}{c}} \leq 2/3$. Thus $\beta_x \leq 1/2(1 + \frac{\Delta_{i'}}{\binom{n}{c}}) \leq 5/6$.

If either 0 or j is in $[\frac{w(x)}{l} - 3, \frac{w(x)}{l} + 3]$, to prove our conclusion, we only need to check the situation that $i' = 0$ and $i' = j$.

If $i' = 0$ or j then,

$$\beta_{x,j} \leq \frac{1}{2}(1 + \frac{\binom{n-l}{j}}{\binom{n}{j}}) \leq \frac{1}{2}(1 + (1 - \frac{l}{n})^j) \leq \frac{1}{2}(1 + (1 - \frac{l}{n})^{0.8n/l}) \leq 3/4.$$

For the second assertion, let's consider the expander graph $G = (V, E)$. Assume v is a random node uniformly drawn from V. For any $i \in [n]$, the conditional random variable $f(v)|_{w(f(v))=i}$ is uniformly distributed on B_i. This is because v is uniform, thus $v'_j, j \in [n]$ are independently identically distributed according to Lemma 9.5. So $\Pr[f(v)|_{w(f(v))=i} = v'], v' \in B_i$ are all equal.

According to the Union Bound,

$$\begin{aligned}|V_x|/|V| &\leq (1 - \Pr[w(f(v)) \in [0.8c, 1.2c]]) + \sum_{i \in [0.8c, 1.2c]} \Pr[w(f(v)) = i]\frac{|A_{x,i}|}{|B_i|} \\ &\leq (1 - \Pr[w(f(v)) \in 0.8c, 1.2c]) + \sum_{i \in [0.8c, 1.2c]} \Pr[w(f(v)) = i] \times 5/6 \\ &\leq 2\exp\{-\Theta(c)\} + 5/6.\end{aligned} \tag{24}$$

Our assertion follows as $c = n/l \geq 2\log n$.

For the 3rd assertion, let's consider $\Pr[MX = 0]$ when $l \leq w(X) \leq n - l$.

By Theorem 9.1, for any x such that $l \leq w(x) \leq n - l$, $\Pr[Mx = \mathbf{0}] \leq (\frac{|V_x|}{|V|} + \lambda)^t$. Here $\frac{|V_x|}{|V|} \leq 5/6 + 1/\mathsf{poly}(n)$.

Let X_1, X_2 be independent random variables and have the same distribution as X.

$$p_0 = \Pr_{X_1, X_2}[MX_1 = MX_2] = \sum_{x_2 \in \mathsf{supp}(X_2)} \Pr[X_2 = x_2] \cdot \Pr[MX_1 = Mx_2]$$

For any fixed $x_2 \in \mathsf{supp}(X_2)$, let $X' = X_1 \oplus x_2$. So $\Pr[MX_1 = Mx_2] = \Pr[MX' = \mathbf{0}]$. We know that X' is also an (n,k)-source. As a result, we have the following.

$$\begin{aligned}
&Pr[M(X') = 0] \\
\leq &\Pr[w(X') \notin [l, n-l]] + \Pr[w(X') \in [l, n-l]] \times \Pr[MX' = \mathbf{0}|_{w(X') \in [l,n-l]}] \\
\leq &\Pr[w(X') \notin [l, n-l]] + \Pr[MX' = \mathbf{0}|_{w(X') \in [l,n-l]}] \\
\leq &\Pr[w(X') \notin [l, n-l]] + (\frac{|V_x|}{|V|} + \lambda)^t \\
\leq &\Pr[w(X') \notin [l, n-l]] + (5/6 + 1/\mathsf{poly}(n) + \lambda)^t \\
\leq &2 \times 2^{-0.5k} + (5/6 + 1/\mathsf{poly}(n) + \lambda)^t.
\end{aligned} \tag{25}$$

Thus,

$$\begin{aligned}
p_0 = &\sum_{x_2 \in \mathsf{supp}(X_2)} \Pr[X_2 = x_2] \cdot \Pr[M(X_1 \oplus x_2) = \mathbf{0}] \\
\leq &2^{-0.5k+1} + (5/6 + 1/\mathsf{poly}(n) + \lambda)^t.
\end{aligned} \tag{26}$$

For the 4th conclusion, let's fix a $M_u \in \mathsf{supp}(M)$. We consider $H_2(M_u X)$ as the following.

$$\begin{aligned}
H_2(M_u X) &= -\log \Pr[M_u X_1 = M_u X_2] \\
&= -\log \sum_{x_1, x_2 \in \mathsf{supp}(X)} \Pr[X_1 = x_1] \Pr[X_2 = x_2] I_{M_u x_1 = M_u x_2}
\end{aligned} \tag{27}$$

Here I_e is the indicator function such that $I_e = 1$ if and only if the event e happens. Here for each x_1, x_2, $I_{M_u x_1 = M_u x_2}$ is a fixed value (either 0 or 1, not a random variable because M_u is fixed).

Next let's consider M to be a random variable generated by the seed U_d.

Let $Z_M = \sum_{x_1, x_2 \in \mathsf{supp}(X)} \Pr[X_1 = x_1] \Pr[X_2 = x_2] I_{Mx_1 = Mx_2}$. We know that

$$\mathbf{E} Z_M = \sum_{x_1, x_2 \in \mathsf{supp}(X)} \Pr[X_1 = x_1] \Pr[X_2 = x_2] \Pr[Mx_1 = Mx_2] = p_0.$$

So according to Markov's Inequality,

$$\Pr[Z_M \geq p_0^{2/3}] \leq p_0^{1/3}.$$

So with probability at least $1 - p_0^{1/3}$ over M (over U_d), $Z_M \leq p_0^{2/3}$.

Let's fix $M_u \in \mathsf{supp}(M)$ such that $Z_{M_u} \leq p_0^{2/3}$.

Thus

$$\begin{aligned}
H_\infty(M_u X) &\geq 1/2 H_2(M_u X) \\
&= 1/2(-\log(Z_{M_u})) \\
&\geq -1/2 \log(p_0^{2/3}) \\
&= \frac{1}{3} \log \frac{1}{p_0}.
\end{aligned} \tag{28}$$

This concludes that $U_d \circ \mathsf{Cond}(X, U_d) = U_d \circ MX$ is $p_0^{1/3}$-close to $U_d \circ W$ where for every u, $W|_{U_d = u}$ has entropy $1/3 \log \frac{1}{p_0}$.

As we know $p_0 \leq 2^{-0.5k+1} + (5/6 + 1/\mathsf{poly}(n) + \lambda)^t$, $k = \Omega(\log^2 n)$. So if $t = 10k$, then $p_0^{1/3} \leq 2^{-0.1k}$.

According to conclusion 5, $U_d \circ \mathsf{Cond}(X, U_d)$ is ϵ_c-close to $U_d \circ W$ where for every u, $W|_{U_d = u}$ has entropy at least $0.1k$ where $\epsilon_c = 2^{-0.1k}$. This proves the last assertion.

□

9.2 Seed Length Reduction

The seed length can be shorter by applying the following PRG technique.

Theorem 9.9 (Space Bounded PRG[36]). *For any $s > 0$, $0 < n \leq 2^s$, there exists an explicit PRG $g : \{0,1\}^r \to \{0,1\}^n$, such that for any algorithm A using space s,*

$$|\Pr[A(g(U_r)) = 1] - \Pr[A(U_n) = 1]| \leq 2^{-s}.$$

Here $r = O(s \log n)$, U_r is uniform over $\{0,1\}^r$, U_n is uniform over $\{0,1\}^n$.

Lemma 9.10. *Let $g : \{0,1\}^{r=O(\log^2 n)} \to \{0,1\}^{r_0=O(n \log n)}$ be the PRG from Lemma 9.9 with error parameter $\epsilon_g = 1/\mathsf{poly}(n)$. Replace the 1-3 steps of Construction 9.6 with the following 3 steps.*

1. *Construct an expander graph $\tilde{G} = (\tilde{V}, \tilde{E})$ where $\tilde{V} = \{0,1\}^r$ and $\lambda = 0.01$.*
2. *Use a uniform random string U_1 of length r to select a vertex $\tilde{v}_1$ of $\tilde{V}$.*
3. *Take a random walk on $\tilde{G}$ starting from $\tilde{v}_1$ to get $\tilde{v}_2, \ldots, \tilde{v}_t$, for $t = 10k$. Let $v_i = g(\tilde{v}_i), i = 1, \ldots, t$.*

Let $\tilde{V}_x = \{v \in \tilde{V} : \langle f(g(v)), x\rangle = 0\}$. We have the following conclusions.

1. *For any $x \in \{0,1\}^n$ such that $w(x) \in [l, n-l]$,*

$$|\frac{|\tilde{V}_x|}{|\tilde{V}|} - \frac{|V_x|}{|V|}| \leq \epsilon_g.$$

2. *$p_0 = \Pr[MX_1 = MX_2] \leq 2^{-0.5k+1} + (5/6 + 1/\mathsf{poly}(n) + \lambda)^t$ where X_1, X_2 are independent random variables that both have the same distribution as X.*
3. *$U_d \circ \mathsf{Cond}(X, U_d)$ is $p_0^{1/3}$-close to $U_d \circ W$. Here U_d is a uniform distribution of length d. For every u, $W|_{U_d=u}$ has entropy $\frac{1}{3} \log \frac{1}{p_0}$.*
4. *$U_d \circ \mathsf{Cond}(X, U_d)$ is $\epsilon_c = 2^{-0.1k}$-close to $U_d \circ W$ where $\forall u \in \{0,1\}^d$, $W|_{U_d=u}$ has entropy $d + 0.1k$.*

Proof. Consider the following algorithm A which decides whether $v \in V_x$ on input (v, x).

Algorithm 9.1: $A(v, x)$

```
Input: v ∈ {0,1}^{r_0} and x ∈ {0,1}^n
res = 0 ;
for i = 1 to n do
    compute f(v)_i ;
    if f(v)_i = 1 then
        res = res + f(v)_i · x_i ;
    end
end
if res = 0 then Output 1 ;
else Output 0 ;
```

We can see that algorithm A runs in space $O(\log n)$ because $f(v)_i, i = 1, \ldots, n$ can be computed sequentially by using $O(\log n)$ space according to Lemma 9.5 and all the other variables need $O(\log n)$ space to record. Also $A(v, x) = 1$ if and only if $v \in V_x$. So $\Pr[A(U_{r_0}, x) = 1] = \frac{|V_x|}{|V|}$.

Similarly, we can see $\Pr[A(g(U_r), x) = 1] = \frac{|\tilde{V}_x|}{|\tilde{V}|}$.

According to the definition of our PRG g, we know that,

$$|\Pr[A(g(U_r), x) = 1] - \Pr[A(U_{r_0}, x) = 1]| \leq \epsilon_g.$$

So

$$|\frac{|\tilde{V}_x|}{|\tilde{V}|} - \frac{|V_x|}{|V|}| \leq \epsilon_g.$$

For the 2nd assertion, let's consider $\Pr[MX = 0]$ when $l \leq w(X) \leq n - l$.

By Theorem 9.1, for any x such that $l \leq w(x) \leq n - l$, $\Pr[Mx = \mathbf{0}] \leq (\frac{|\tilde{V}_x|}{|\tilde{V}|} + \lambda)^t$.

Let X_1, X_2 be independent random variables and have the same distribution as X.

$$p_0 = \sum_{x_2 \in \mathsf{supp}(X_2)} \Pr[X_2 = x_2] \cdot \Pr[MX_1 = Mx_2]$$

For any fixed $x_2 \in \mathsf{supp}(X_2)$, let $X' = X_1 \oplus x_2$. So $\Pr[MX_1 = Mx_2] = \Pr[MX' = \mathbf{0}]$. We know that X' is also an (n, k)-source. As a result, we have the following.

$$\begin{aligned}
&\Pr[M(X') = 0] \\
\leq &\Pr[w(X') \notin [l, n-l]] + \Pr[w(X') \in [l, n-l]] \times \Pr[MX' = \mathbf{0}|_{w(X') \in [l, n-l]}] \\
\leq &\Pr[w(X') \notin [l, n-l]] + \Pr[MX' = \mathbf{0}|_{w(X') \in [l, n-l]}] \\
\leq &\Pr[w(X') \notin [l, n-l]] + (\frac{|\tilde{V}_x|}{|\tilde{V}|} + \lambda)^t \\
\leq &\Pr[w(X') \notin [l, n-l]] + (\frac{|V_x|}{|V|} + \epsilon_g + \lambda)^t \\
\leq &\Pr[w(X') \notin [l, n-l]] + (5/6 + 1/\mathsf{poly}(n) + \lambda)^t \\
\leq &2 \times 2^{-0.5k} + (5/6 + 1/\mathsf{poly}(n) + \lambda)^t.
\end{aligned} \tag{29}$$

Thus,

$$\begin{aligned}
p_0 = &\sum_{x_2 \in \mathsf{supp}(X_2)} \Pr[X_2 = x_2] \cdot \Pr[M(X_1 \oplus x_2) = \mathbf{0}] \\
\leq &2^{-0.5k+1} + (5/6 + 1/\mathsf{poly}(n) + \lambda)^t.
\end{aligned} \tag{30}$$

Conclusion 3 and 4 follow the same proof as that of Lemma 9.8.

□

Theorem 9.11. *For any $k = \Omega(\log^2 n)$, there exists an explicit construction of an $(n, k, 10k, 0.1k, 2^{-0.1k})$-condenser with seed length $\Theta(k)$.*

Proof. According to Lemma 9.10, it immediately follows that the function Cond in Construction 9.6 is an $(n, k, t, 0.1k, \epsilon_c)$-condenser for $t = 10k$, $\epsilon_c = 2^{-0.1k}$.

Now consider the seed length. We know that $|U_1| = \Theta(\log^2 n)$. For the random walks, the random bits needed have length $\Theta(t) = \Theta(k)$. So the seed length is $|U_1| + \Theta(t) = \Theta(k)$. □

9.3 Locality Control

There is one problem left in our construction. We want the locality of our extractor to be small. However, in the current construction, we cannot guarantee that the locality is small, because the random walk may hit some vectors that have large weights. We bypass this barrier by setting these vectors to be $\mathbf{0}$.

We need the following Chernoff Bound for random walks on expander graphs.

Theorem 9.12 ([22] Theorem 1)**.** *Let G be a regular graph with N vertices where the second largest eigenvalue is λ. For every $i \in [t]$, let $f_i : [N] \to [0,1]$ be any function. Consider a random walk $v_1, v_2, \ldots, v_t$ in G from a uniform start-vertex v_1. Then for any $\epsilon > 0$,*

$$\Pr[|\sum_{i=1}^{t} f_i(v_i) - \sum_{i=1}^{t} \mathbf{E} f_i| \geq \epsilon t] \leq 2e^{-\frac{\epsilon^2(1-\lambda)t}{4}}.$$

Now we give our final construction.

Construction 9.13. *For any $k = \Omega(\log^2 n)$, we construct an $(n, k, t = 10k, 0.08k, \epsilon_c)$-condenser $\mathsf{Cond} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^t$ with $d = \Theta(k)$, locality $c = n/l, l = \frac{k}{2\log n}$, $\epsilon_c = 2^{-k/500000}$.*

Let $g : \{0,1\}^{r=O(\log^2 n)} \to \{0,1\}^{r_0=O(n\log n)}$ be the PRG from Lemma 9.9 with error parameter $\epsilon_g = 1/\mathsf{poly}(n)$.

1. *Construct an expander graph $\tilde{G} = (\tilde{V}, \tilde{E})$ where $\tilde{V} = \{0,1\}^r$ and $\lambda(\tilde{G}) = 0.01$ where $r = O(\log^2 n)$.*
2. *Using a uniform random string U_1 of length r to select a vertex $\tilde{v}_1$ of $\tilde{V}$.*
3. *Take a random walk on $\tilde{G}$ to get $\tilde{v}_2, \ldots, \tilde{v}_t$. Let $v_i = g(\tilde{v}_i), i = 1, \ldots, t$.*
4. *For $i \in [t]$, get an n-bit string $v'_i = f(v_i)$ such that $\forall j \in [n], \Pr[v'_{i,j} = 1] - 1/l| \leq 1/n^2$, where $f : \{0,1\}^{r_0} \to \{0,1\}^n$ follows from Lemma 9.5, $r_0 = O(n \log n)$.*
5. *Let $M = (v'_1, \ldots, v'_t)^T$.*
6. *Construct the matrix $M' = (\bar{v}_1, \ldots, \bar{v}_t)$ such that for $i \in [t]$, if $w(v'_i) > 1.2c$, $\bar{v}_i = \mathbf{0}$, otherwise $\bar{v}_i = v'_i$.*
7. *Let $\mathsf{Cond}(x, u) = M'x$.*

Let $\tilde{T} = \{v \in \tilde{V} : w(f(g(v))) \in [0.8c, 1.2c]\}$.

Lemma 9.14. *In Construction 9.13, with probability $1 - 2e^{-k/500000}$,*

$$|\{i : w(v'_i) \in [0.8c, 1.2c]\}| \geq 0.998t.$$

Proof. Consider the following algorithm A. Given $v \in \{0,1\}^{r_0}$, A tests whether $w(f(v)) \in [0.8c, 1.2c]$.

Algorithm 9.2: $A(v)$

Input: $v \in \{0,1\}^{r_0}$
count = 0 ;
for $i = 1$ *to* n **do**
  compute $f(v)_i$;
  if $f(v)_i = 1$ **then**
    count++;
  end
end
if *count is in* $[0.8c, 1.2c]$ **then** Output 1 ;
else Output 0 ;

It can be seen that A runs in space $O(\log n)$. Because $f(v)_i, i = 1, \ldots, n$ can be computed sequentially using space $O(\log n)$ according to Lemma 9.5. Also all the iterators and variables used during the computation require only $O(\log n)$ space.

As a result, according to the definition of space bounded PRG, for any $\epsilon_g = 1/\mathsf{poly}(n)$,

$$|\Pr[A(g(U_r)) = 1] - \Pr[A(U_{r_0}) = 1]| \leq \epsilon_g.$$

We know that $\Pr[A(g(U_r)) = 1] = \frac{|\tilde{T}|}{|\tilde{V}|}$ and $\Pr[A(U_{r_0}) = 1] = \frac{|T|}{|V|}$. Thus, $|\frac{|\tilde{T}|}{|\tilde{V}|} - \frac{|T|}{|V|}| \leq \epsilon_g$.

According to Lemma 9.7, $\frac{|T|}{|V|} \geq 1 - 2\exp\{-\Theta(c)\}$.

As a result, $\frac{|\tilde{T}|}{|\tilde{V}|} \geq 1 - 2\exp\{-\Theta(c)\} - \epsilon_g \geq 1 - 1/\mathsf{poly}(n)$.

Thus for each i, $\Pr[w(f(g(\tilde{v}))) \in [0.8c, 1.2c]] \geq 1 - 1/\mathsf{poly}(n)$. Let $\mathbf{E} I_{\tilde{v}_i \in \tilde{T}} = u$. Then $u \geq 1 - 1/\mathsf{poly}(n)$.

According to our construction, we can assume $I_{\tilde{v}_i \in \tilde{T}} = h(\tilde{v}_i), i = 1, \ldots, t$ for some function h. By Theorem 9.12,

$$\Pr[|\sum_{i=1}^{t} I_{\tilde{v}_i \in \tilde{T}} - \sum_{i=1}^{t} u| \geq 0.001t] \leq 2e^{-\frac{(0.001)^2(1-\lambda)t}{4}} \leq 2e^{-t/5000000}$$

We know that $t = 10k$ and $|\{i : w(v'_i) \in [0.8c, 1.2c]\}| = \sum_{i=1}^{t} I_{\tilde{v}_i \in \tilde{T}}$. So with probability at least $1 - 2e^{-k/500000}$,

$$|\{i : w(v'_i) \in [0.8c, 1.2c]\}| \geq \sum_{i=1}^{t} u - 0.001t \geq (1 - 1/\mathsf{poly}(n))t - 0.001t \geq 0.998t.$$

□

Lemma 9.15. *The function* $\mathsf{Cond} : \{0,1\}^n \times \{0,1\}^d$ *in Construction 9.13 is an* $(n, k, t, 0.08k, \epsilon_c)$*-condenser with seed length* $\Theta(k)$.

Proof. According to Lemma 9.11, we know that for $\epsilon = 2^{-0.1k}$, $U_d \circ MX$ is ϵ-close to $U_d \circ W$ where for every $a \in \{0,1\}^d$, $H_\infty(W|_{U_d=a}) = 0.1k$. Let $M'X = h(U_d, MX)$. According to our construction, we know that h is a deterministic function. More specifically, $h(u, y)$ will set the ith coordinate of y to be 0 for any i such that $\tilde{v}_i \notin \tilde{T}$. The function h can check $\tilde{v}_i \notin \tilde{T}$ according to u deterministically.

As a result,

$$\begin{aligned}&\mathsf{SD}(U_d \circ M'X, U_d \circ h(U_d, W))\\ =&\mathsf{SD}(U_d \circ h(U_d, MX), U_d \circ h(U_d, W))\\ \leq&\mathsf{SD}(U_d \circ MX, U_d \circ W)\\ \leq&\epsilon.\end{aligned} \tag{31}$$

Now let's consider the entropy of $U_d \circ h(U_d, W)$. let $\epsilon_0 = 2e^{-k/500000}$. By Lemma 9.14, for $1-\epsilon_0$ fraction of $u \in \{0,1\}^d$, there are at most $0.002t$ bits in $W|_{U_d=u}$ that are set to be 0.

As a result, for $1-\epsilon_0$ fraction of $u \in \{0,1\}^d$, $u \circ h(u, W|_{U_d=u})$ has entropy $0.1k - 0.002t \geq 0.08k$. As a result, $U_d \circ h(U_d, W)$ is ϵ_0-close to $U_d \circ W'$ where for every $u \in \{0,1\}^d$, $W'|_{U_d=u}$ has entropy $0.08k$. So $U_d \circ M'X$ is $\epsilon + \epsilon_0 \leq 2^{-k/500000}$-close to $U_d \circ W'$ where for every $u \in \{0,1\}^d$, $W'|_{U_d=u}$ has entropy $0.08k$. □

Lemma 9.16. *The locality of Construction 9.13 is* $1.2c = \Theta(\frac{n}{k}\log n)$.

Proof. As for every M'_i, the number of 1s in it is at most $1.2c$, the locality is $1.2c = 1.2n/l = \Theta(\frac{n}{k}\log n)$ □

Theorem 9.17. *For any* $k = \Omega(\log^2 n)$*, there exists an* $(n, k, t = 10k, 0.08k, \epsilon_c)$*-condenser* $\mathsf{Cond} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^t$ *with* $d = \Theta(k)$ *,* $\epsilon_c = 2^{-k/500000}$ *and the locality is* $\Theta(\frac{n}{k}\log n)$.

Proof. It follows from Lemma 9.15 and Lemma 9.16.

□

Theorem 9.18. *For any* $k = \Omega(\log^2 n)$*, for any constant* $\gamma \in (0,1)$*, there exists a strong* (k,ϵ)*-extractor* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$*, where* ϵ *can be as small as* $2^{-k^{\Omega(1)}}$*,* $d = \Theta(k), m = (1-\gamma)k$ *and the locality is* $\frac{n}{k}\log^2(1/\epsilon)(\log n)\mathsf{poly}(\log k)$.

Proof Sketch. We combine our $(n, k, m_c = 10k, 0.08k, \epsilon_c = 2^{-0.1k})$-condenser $\mathsf{Cond} : \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^{m_c}$ where $r = \Theta(k)$ from Lemma 9.11 with the $(0.08k, \epsilon_0)$-extractor $\mathsf{Ext}_0 : \{0,1\}^{m_c} \times \{0,1\}^{d_0 = k^\alpha} \to \{0,1\}^{m_0}$ from Theorem 7.5 for some constant $\alpha \in (0,1]$ and for $\epsilon_c = 2^{-k^{\Omega(1)}}$.

Let $\mathsf{Ext}(X,U) = \mathsf{Ext}_0(\mathsf{Cond}(X, U_1), U_2)$, where $U = U_1 \circ U_2$. We know that $U \circ \mathsf{Ext}(X,U)$ is $\epsilon = \epsilon_c + \epsilon_0 = 2^{-k^{\Omega(1)}}$-close to uniform distribution over $\{0,1\}^{\Theta(k)}$.

The locality of Cond is $\Theta(\frac{n}{k}\log n)$. The locality of Ext_0 is $\log^2(1/\epsilon_0)\mathsf{poly}(\log k)$. So the overall locality is $\frac{n}{k}\log^2(1/\epsilon)(\log n)\mathsf{poly}(\log k)$. The seed length is $|U| = |U_1| + |U_2| = d_0 + \Theta(k) = \Theta(k)$.

Our theorem holds by applying the extraction in parallel technique in Lemma 6.4 to increase the output length to $(1-\gamma)k$. □

10 Applications

In this section, we give some constructions of PRGs based on our AC0 extractor.

10.1 PRG in AC0 Based on Random Local One-way Function

Our first construction is based on random local one-way functions following the method of Applebaum [2].

Let $\mathsf{dist}(\cdot)$ denotes the hamming distance between the two input strings (with equal length).

Definition 10.1 (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.*

Remark 10.2. *Here we do not require the indices $i_0, i_1, \ldots, i_{d-1}$ to be distinct. This setting is the same as that in [9] and [18] (Random Construction).*

Definition 10.3 (Predicate)**.** *A d-ary predicate $Q : \{0,1\}^d \to \{0,1\}$ is a function which partitions $\{0,1\}^d$ in to V_0 and V_1, where $V_a = \{w \in \{0,1\}^d | Q(w) = a\}$ for $a = 0, 1$.*

Let $H_Q = (V_0 \cup V_1, E)$ be a bipartite graph where $(u, v) \in V_0 \times V_1$ is an edge if $\mathsf{dist}(u, v) = 1$. Let $\mathcal{M}$ be all the possible matchings of H_Q. The size of the maximum matching of H_Q is

$$\mathsf{Match}(Q) = \max_{M \in \mathcal{M}} \Pr_v[\exists u, (u, v) \in M \text{ or } (v, u) \in M] = \max_{M \in \mathcal{M}} 2|M|/2^d,$$

where v is uniformly distributed in $V_0 \cup V_1$.

Definition 10.4 (Collection of Functions)**.** *For $s = s(n), m = m(n)$, a collection of functions $F : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^m$ takes an input (k, x) and outputs $F(k, x)$. Here k is a public index and x can be viewed as the input for the kth function in the collection. We also denote $F(k, x)$ as $F_k(x)$ where F_k is the kth function in the collection.*

Remark 10.5. *For simplicity, we usually consider n as an exponential of 2.*

In the following paragraph, an efficient adversary is defined to be a probabilistic polynomial time Turing Machine. Also the term efficient means in probabilistic polynomial time.

Definition 10.6 (Approximate One-way Function for Collection of Functions)**.** *For $\delta = \delta(n) \in (0, 1)$ and $\epsilon = \epsilon(n) \in (0, 1)$, a collection of functions $F : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^m$ is an (ϵ, δ)-approximate 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*

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

where k and x are independent and uniform. Specially, when $\delta = 0$, we say the collection F is ϵ-one-way.

Definition 10.7 (Goldreich's Random Local Function [18])**.** *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 $(G, x) \to f_{G,Q}(x)$.

Lemma 10.8. *For every $d = O(\log n)$, every $m = \mathsf{poly}(n)$ and every predicate $Q : \{0,1\}^d \to \{0,1\}$, the random local function $F_{Q,n,m}$, following Definition 10.7, is in AC^0.*

Proof. For every $i \in [m]$, we claim that the ith output bit of $F_{Q,n,m}(G, x)$ can be computed in AC^0. The reason is as follows. We know that $F_{Q,n,m}(G, x)_i = Q(x_{S_i})$. So it is determined by d bits of x and S_i which corresponds to $d \log n$ bits of G. Thus for $S_i = (j_0, j_1, \ldots, j_{d-1})$, $\forall l \in [d]$, the lth input for Q is

$$x_{j_l} = \bigvee_{k=0}^{n} (I_{j_l=k} \wedge x_k) = \bigwedge_{k=0}^{n} (I_{j_l \neq k} \vee x_k).$$

As $|j_l| = \log n$, $I_{j_l=k}$ and $I_{j_l \neq k}$ can be computed in AC^0 by Lemma 2.11. So every input bit in x_{S_i} can be computed in AC^0. As $d = O(\log n)$, we know that $F_{Q,n,m}(G,x)_i = Q(x_{S_i})$ can be computed in AC^0. Thus $F_{Q,n,m}(G,x)$ can be computed in AC^0.

□

Definition 10.9. *Two distribution ensembles $Y = \{Y_n\}$ and $Z = \{Z_n\}$ are ϵ-indistinguishable if for every efficient adversary A,*

$$|\Pr[A(1^n, Y_n) = 1] - \Pr[A(1^n, Z_n) = 1]| \leq \epsilon(n).$$

Here the subscript of a random variable indicates its length.

Definition 10.10 (PRG for a Collection of Functions)**.** *Let $m = m(n)$. A collection of functions $F : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^m$ is an ϵ-PRG, if $(K, F_K(U_n))$ is ϵ-indistinguishable from the uniform distribution. Here K is uniform over $\{0,1\}^s$, U_n is uniform over $\{0,1\}^n$.*

A collection of functions $F : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^m$ is ϵ-unpredictable generator (UG) if for every efficient adversary A and every sequence of indices $\{i_n\}_{n\in\mathbb{N}}$ where $i_n \in [m(n)]$, we have that

$$\Pr_{k \leftarrow U_s, x \leftarrow U_n}[A(k, F_k(x)_{[0,\ldots,i_n-1]}) = F_k(x)_{i_n}] \leq \epsilon(n)$$

for sufficiently large n's. Here F is ϵ-last-bit unpredictable generator (LUG) if $i_n = m(n) - 1$.

Remark 10.11. *Let $t = t(r)$. A function $G : \{0,1\}^r \to \{0,1\}^t$ is a classic ϵ-PRG, if $(K, F_K(U_n))$ is ϵ-indistinguishable from the uniform distribution. Here K is uniform over $\{0,1\}^r$, U_n is uniform over $\{0,1\}^n$.*

The definition of PRG for a collection of functions implies the classic definition of PRG. Following our definition, if there exists an explicit ϵ-PRG $F(\cdot,\cdot)$ for a collection of functions, we know $(U_s, F_{U_s}(U_n))$ is ϵ-indistinguishable from uniform distributions. Let $G : \{0,1\}^{r=s+n} \to \{0,1\}^{t=s+m}$ be such that $\forall k \in \{0,1\}^s, \forall x \in \{0,1\}^n, G(k \circ x) = k \circ F(k,x)$. We know that $G(U_r)$ is indistinguishable from uniform distributions. So G is a classic ϵ-PRG.

Definition 10.12. *An ϵ-LPRG is an ϵ-PRG whose output length is linear of its input length (including the index length, $m > (1+\delta)(n+s)$ for some constant δ).*

An ϵ-PPRG is an ϵ-PRG whose output length is a polynomial of its input length (including the index length, $m > (n+s)^{(1+\delta)}$ for some constant δ).

Lemma 10.13. *For every $c \in \mathbb{N}^+$, an ϵ-PRG $G : \{0,1\}^r \to \{0,1\}^t$ in AC^0 can be transformed to an $(c\epsilon)$-PRG $G' : \{0,1\}^r \to \{0,1\}^{t(t/r)^c}$.*

Here $G'(\cdot) = G^{(c)}(\cdot)$, where $G^{(i+1)}(\cdot) = G^{(i)}(\cdot), \forall i \in \mathbb{N}^+$ and $G^{(1)}(\cdot) = G(\cdot)$.

If c is a constant, then G' is in AC^0.

Proof. We use inductions. Assume the the output length for $G^{(i)}$ is $t^{(i)}$.

For the basic step, as G is an ϵ-PRG, $G^{(1)} = G$ is an ϵ-PRG.

For the induction step, assume for i, $G^{(i)}$ is an $(i\epsilon)$-PRG. Suppose there exists an efficient adversary A such that

$$|\Pr[A(G^{(i+1)}(U_r)) = 1] - \Pr[A(U_{t^{(i+1)}}) = 1]| > (i+1)\epsilon.$$

We know that

$$\begin{aligned} &|\Pr[A(G^{(i+1)}(U_r)) = 1] - \Pr[A(U_{t^{(i+1)}}) = 1]| \\ \leq &|\Pr[A(G^{(i+1)}(U_r)) = 1] - \Pr[A(G(U_{t^{(i)}})) = 1]| + |\Pr[A(G(U_{t^{(i)}})) = 1] - \Pr[A(U_{t^{(i+1)}}) = 1]| \end{aligned} \tag{32}$$

As $|\Pr[A(G(U_{t^{(i)}})) = 1] - \Pr[A(U_{t^{(i+1)}}) = 1]| \leq \epsilon$,

$$|\Pr[A(G^{(i+1)}(U_r)) = 1] - \Pr[A(G(U_{t^{(i)}})) = 1]| > i\epsilon$$

contradicting the the induction assumption. So $G^{(i+1)}$ is an $(i+1)\epsilon$-PRG. □

Theorem 10.14. *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. *For some constant $c = c(d) > 1$, if $m > cn$, then there exists a ϵ-LPRG in AC^0 with ϵ being negligible.*
2. *For any constant $c > 1$, if $m > n^c$, then there exists a ϵ-PPRG in AC^0 with ϵ being negligible.*

Before we prove Theorem 10.14, we first use it to obtain our main theorem in this subsection.

Theorem 10.15. *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. *For some constant $c > 1$, if $m > cn$, then for any constant $a > 1$, there exists a ϵ-LPRG $G : \{0,1\}^r \to \{0,1\}^t$ in AC^0, where $t \geq ar$ and ϵ is negligible.*
2. *For any constant $c > 1$, if $m > n^c$, then for any constant $a > 1$ there exists a ϵ-PPRG $G : \{0,1\}^r \to \{0,1\}^t$ in AC^0, where $t \geq r^a$ and ϵ is negligible.*

Proof. For the first assertion, let the LPRG in Theorem 10.14 be $G_0 : \{0,1\}^{r_0} \to \{0,1\}^{t_0}$ with $t_0 > c_0 r_0$ for some constant $c_0 > 1$. We apply the construction in Lemma 10.13 to obtain $G^{(c_1)}$ such that $c_0^{c_1} \geq a$. So c_1 is a constant. By Lemma 10.13 we know that $G^{(c_1)}$ is a $c_1\epsilon$-PRG in AC^0. This proves the first assertion.

By the same reason, the second assertion also holds. □

Construction 10.16. *Let $F_{Q,n,m} : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^m$ be the random local function following Definition 10.7. We construct $F' : \{0,1\}^s \times \{0,1\}^{n'} \to \{0,1\}^{m'}$ where $n' = tn, m' = tm, t = n$.*

1. *Draw G uniformly from $\{0,1\}^s$.*
2. *Draw $x^{(1)}, x^{(2)}, \ldots, x^{(t)}$ independently uniformly from $\{0,1\}^n$. Let $x = (x^{(1)}, x^{(2)}, \ldots, x^{(t)})$.*
3. *Output $F'(G,x) = \bigcirc_{i=1}^t G(x^{(i)})$.*

Lemma 10.17. *In Construction 10.16, for every constant $d \in \mathbb{N}^+$, every predicate $Q : \{0,1\}^d \to \{0,1\}$, every $m = \mathsf{poly}(n)$ and every $\epsilon = 1/\mathsf{poly}(n)$, if $F_{Q,n,m}$ is $(\frac{1}{2} + \epsilon)$-last-bit unpredictable then F' is $(\frac{1}{2} + \epsilon(1 + 1/n))$-unpredictable.*

Proof. Suppose there exists a next-bit predictor P and a sequence of indices $\{i_n\}$ such that

$$\Pr_{x \leftarrow U_n, G \leftarrow U_s, y = F'(G,x)}[P(G, y_{0,\ldots,i_n-1}) = y_{i_n}] \geq \frac{1}{2} + \epsilon(n)(1 + 1/n)$$

for sufficiently large n's.

Now we construct a last-bit predicator P' which can predicate the last bit of $F_{Q,n,m}$ with success probability $1/2 + \epsilon$.

By Remark 3.2 of [2], P' can find an index $j \in [m']$ by running a randomized algorithm M in polynomial time, such that, with probability $1 - 2^{-\Theta(n)}$ over the random bits used in M,

$$\Pr_{G,x,y=F'(G,x)}[P(G, y_{[0,\ldots,j-1]}) = y_j] > \frac{1}{2} + \epsilon(n) + \frac{\epsilon(n)}{2n}.$$

Recall that in Remark 3.2 of [2], M' tries every index and pick the best one.

According to Construction 10.16, assume $j = an + b$ for some $a, b \in \mathbb{N}, b < n$.

Given $(G, y_{[0,\ldots,m-2]})$, P' generates $x^{(1)}, x^{(2)}, \ldots, x^{(a-1)}$ independently uniformly over $\{0,1\}^n$. Also P' constructs a hypergraph G' by swapping S_b and S_{m-1} of G. Next, P' computes $y' = \bigcirc_{i=1}^{a} G'(x^{(i)}) \circ y_{[0,\ldots,b-2]}$. Finally P' outputs $P(G', y')$. As G is uniform, G' is also uniform. Also as $x^{(1)}, \ldots, x^{(a)}$ are uniform, (G', y') has the same distribution as $(G, y_{[0,\ldots,j-1]})$. So

$$\Pr[P'(G', y') = y_{m-1}] \geq \frac{1}{2} + \epsilon(n) + \frac{\epsilon(n)}{2n} - 2^{\Theta(n)} > \frac{1}{2} + \epsilon(n).$$

This contradicts that $F_{Q,n,m}$ is $(\frac{1}{2} + \epsilon)$-last-bit unpredictable. □

Theorem 10.18 ([2], Section 5)**.** *For every constant $d \in \mathbb{N}$, predicate $Q : \{0,1\}^d \to \{0,1\}$, and constant $\epsilon \in (0, \mathsf{Match}(Q)/2)$, there exists a constant $c > 0$ such that for every polynomial $m > cn$ the following holds. If the collection $F_{Q,n,m}$ is $\epsilon/5$-one-way then it is a $(1 - \mathsf{Match}(Q)/2 + \delta)$-last-bit UG where $\delta = \epsilon(1 - o(1))$. Thus it is also a $(1 - \mathsf{Match}(Q)/2 + \epsilon)$-UG.*

Remark 10.19. *Our definition of random local function has only one difference with the definition of [2]. That is, for each hyperedge we do not require the incoming vertices to be distinct. This difference does not affect the correctness of Theorem 10.18.*

Construction 10.20 (Modified from [2] Construction 6.8)**.** *Let $F : \{0,1\}^{s(n)} \times \{0,1\}^n \to \{0,1\}^{m(n)}$ be a UG and $\mathsf{Ext} : \{0,1\}^{n_1} \times \{0,1\}^{d_1} \to \{0,1\}^{m_1}$ be a strong $(k = \alpha n_1, \epsilon_1)$-extractor following Theorem 6.15 where $n_1 = n$, α is some constant, $\epsilon = 1/2^{\Theta(\log^a n)}$ for some large enough constant $a \in \mathbb{N}^+$, $d_1 = (\log a)^{\Theta(a)}$, $m = 0.9k$.*

We construct the following UG $H : \{0,1\}^{sn} \times \{0,1\}^{n^2 + d_1 n} \to \{0,1\}^{mn}$.

1. *Index: Generate $G_0, G_1, \ldots, G_{n-1}$ independently uniformly over $\{0,1\}^s$. Generate extractor seeds $u_0, u_1, \ldots, u_{m-1}$ independently uniformly over $\{0,1\}^{d_1}$. Denote $G = (G_0, G_1, \ldots, G_{n-1})$ and $u = (u_0, u_1, \ldots, u_{m-1})$.*
2. *Input: Generate $x^{(0)}, x^{(1)}, \ldots, x^{(n-1)}$ independently uniformly over $\{0,1\}^n$.*

   Denote $x = (x^{(0)}, x^{(1)}, \ldots, x^{(n-1)})$.
3. *Output: Compute the $n \times m$ matrix Y whose ith row is $G_i(x^{(i)})$. Let Y_i denote the ith column of Y. Output $H(G, u, x) = \mathsf{Ext}(Y_0, u_0) \circ \mathsf{Ext}(Y_1, u_1) \circ \cdots \circ \mathsf{Ext}(Y_{m-1}, u_{m-1})$.*

Remark 10.21. *There are 2 differences between our construction and Construction 6.8 of [2]. First, we use our AC0-extractor to do extraction. As our extractor is strong, its seed can also be regarded as part of the public key (index). Second, our construction is for any m, while their construction only considers m as a linear function of n.*

Lemma 10.22. *For any constant $\epsilon \in [0, 1/2)$, if F is $(\frac{1}{2} + \epsilon)$-unpredictable, then the mapping H is a PRG with negligible error.*

Proof Sketch. The proof is almost the same as that of Lemma 6.9 of [2].

By the same argument of Lemma 6.9 of [2], we know that for every sequence of efficiently computable index family $\{i_n\}$ and every efficient adversary A, there exists a random variable $W \in \{0,1\}^n$ jointly distributed with G and Y such that

- the min-entropy of W, given any fixed G and the first i_n columns of Y, is at least $n(1-2\epsilon-o(1))$.
- A cannot distinguish between $(G, Y_{[0,\ldots,i_n]})$ and $(G, [Y_{[0,\ldots,i_n-1]}W])$ with more than negligible advantage even when A is given an oracle which samples the distribution (G, Y, W). Here $[Y_{[0,\ldots,i_n-1]}W]$ is a matrix such that the first i_n columns are $Y_{[0,\ldots,i_n-1]}$ and the last column is W.

By the definition of strong extractors, for every family $\{i_n\}$, the distribution

$$(G, u, Y_{[0,\ldots,i_n-1]}, \mathsf{Ext}(Y_{i_n}, u_{i_n}))$$

is indistinguishable from $(G, u, Y_{[1,\ldots,i_n-1]}, U_{m_1})$. Otherwise, suppose there is an adversary B that can distinguish the two distributions. We construct another adversary A as the follows. First A generates a uniform u as seeds for the extractors and invokes B on $(G, u, y, \mathsf{Ext}(v, u))$ where G is generated from uniform, y is drawn from $Y_{[0,\ldots,i_n-1]}$. If v is drawn from Y_{i_n} then B gets a sample from $(G, u, Y_{[0,\ldots,i_n-1]}, \mathsf{Ext}(Y_{i_n}, u))$. If v is drawn from W, then B gets a sample from $(G, u, Y_{[0,\ldots,i_n-1]}, \mathsf{Ext}(W, u))$ which is ϵ_0-close to $(G, u, Y_{[0,\ldots,i_n-1]}, U_{m_1})$ by the definition of strong extractors, where ϵ_0 is negligible according to our settings in Construction 10.20 and U_{m_1} is the uniform distribution of length m_1. So A can distinguish $(G, Y_{[0,\ldots,i_n]})$ and $(G, [Y_{[0,\ldots,i_n-1]}W])$, having the same distinguishing advantage as B does (up to a negligible loss). This is a contradiction.

As a result, for every family $\{i_n\}$, the distributions

$$(G, u, H(G, u, x)_{[0,\ldots,i_n]}) \text{ and } (G, u, H(G, u, x)_{[0,i_n-1]} \circ U_{m_1})$$

are indistinguishable. So H is a $(1/2 + \mathsf{neg}(n))$-UG. By Fact 6.1 (Yao's theorem) of [2], H is a PRG. □

Proof of Theorem 10.14. We combine Construction 10.16 and Construction 10.20 together by using the UG of Construction 10.16 in Construction 10.20. By Theorem 10.18 and Lemma 10.22, we know that our construction gives a PRG (with negligible error). Assume the PRG is $H : \{0,1\}^{s_H} \times \{0,1\}^{n_H} \to \{0,1\}^{m_H}$.

Next we mainly focus on the stretch. The output length of H is $m_H = \Theta(ntm)$, the input length (including the index length) is $s_H + n_H = sn + d_1 n + n^2 t$. Here we know that $s = m \log n$, $t = n$.

Assume $m > cn$ for some constant $c > 1$. We know that $\frac{m_H}{s_H + n_H} = c' > 1$, for some constant c'.

For the polynomial stretch case, assume $m > n^c$ for some constant $c > 1$. We know that $m_H \geq (s_H + n_H)^{c'}$ for some constant $c' > 1$.

For both cases, the construction is in AC^0. The reason is as follows. By Lemma 10.8, the random local function is in AC^0. In Construction 10.16 and Construction 10.20, we compute $O(nt)$ random local functions (some of them share the same index) in parallel. Also our extractor is in AC^0. So the overall construction is in AC^0.

This proves the theorem.

□

10.2 PRG in AC0 for Space Bounded Computation

In this subsection, we give an AC^0 version of the PRG in [38].

Theorem 10.23. *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.

Proof Sketch. We modify the construction of [38] by replacing their extractor with the extractor from Theorem 6.15 for some constant entropy rate and with error parameter $\epsilon' = 2^{-\Theta(\log^c s)}$. In the PRG construction of [38], it only requires an extractor for constant entropy rate. As our extractor meets their requirement, the proof in [38] still holds under this modification.

For the security parameter ϵ, according to [38], $\epsilon = \mathsf{poly}(s)(\epsilon' + 2^{-s})$. As a result, $\epsilon = 2^{-\Theta(\log^c s)}$.

□

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 nc 0. *Computational Complexity*, 17(1):38–69, 2008.

[5] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[6] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. Bpp has subexponential simulation unless Exptime has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

[7] Boaz Barak, Russell Impagliazzo, and Avi Wigderson. Extracting randomness using few independent sources. *SIAM Journal on Computing*, 36(4):1095–1118, 2006.

[8] 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.

[9] Andrej Bogdanov and Youming Qiao. On the security of goldreichs one-way function. *computational complexity*, 21(1):83–127, 2012.

[10] 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, May 2000.

[11] J. L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.

[12] 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.

[13] Anindya De and Luca Trevisan. Extractors using hardness amplification. In *RANDOM 2009, 13th International Workshop on Randomization and Approximation Techniques in Computer Science*, 2009.

[14] 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.

[15] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 25–32, 1989.

[16] O. Goldreich and A. Wigderson. Tiny families of families with random properties: A quality-size trade-off for hashing. *Random Structures and Algorithms*, 11:315–343, 1997.

[17] Oded Goldreich. Three XOR-lemmas - an exposition. *Electronic Colloquium on Computational Complexity (ECCC)*, 2(56), 1995.

[18] 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.

[19] 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.

[20] Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes. *Journal of the ACM*, 56(4), 2009.

[21] 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.

[22] Alexander D Healy. Randomness-efficient sampling within nc1. *Computational Complexity*, 17(1):3–37, 2008.

[23] Johan Hstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Construction of a pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28:12–24, 1993.

[24] 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.

[25] 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.

[26] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 248–253, 1989.

[27] 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.

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

[29] Stasys Jukna. *Extremal combinatorics: with applications in computer science*. Springer Science & Business Media, 2011.

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

[31] 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.

[32] 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.

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

[34] Raghu Meka. Explicit resilient functions matching Ajtai-Linial. *CoRR*, abs/1509.00092, 2015.

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

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

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

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

[39] 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.

[40] Ran Raz, Omer Reingold, and Salil Vadhan. Extracting all the randomness and reducing the error in trevisan's extractors. *JCSS*, 65(1):97–128, 2002.

[41] Maciej Skórski. Shannon entropy versus renyi entropy from a cryptographic viewpoint. In *Cryptography and Coding*, pages 257–274. Springer, 2015.

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

[43] 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.

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

[45] Salil P Vadhan. *Pseudorandomness*. Now, 2012.

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

[47] 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.

[48] Ingo Wegener et al. *The complexity of Boolean functions*, volume 1. BG Teubner Stuttgart, 1987.

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

A Proof of Lemma 4.5

Lemma A.1 (Lemma 4.5 restated, implicit in [25]). *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 $l' = |a| + |s| + |v_1| + |w| = \Theta(l)$.

In order to clearly compute the circuit size, we need the following theorem.

Theorem A.2 (Circuit Size of Majority Function [48] Page 76, Theorem 4.1). *The circuit size of the majority function on n input bits is $O(n)$.*

We need the following version of Goldreich Levin Theorem.

Theorem A.3 (Goldreich-Levin Algorithm [15], Circuit Version). *For any $\epsilon > 0$, for any function $f : \{0,1\}^l \to \{0,1\}^t$, if there is a circuit $C : \{0,1\}^{l+t} \to \{0,1\}$ of size g such that we have*

$$\Pr_{x,r}[C(x, r) = \langle f(x), r \rangle] \geq 1/2 + \epsilon,$$

where r is uniformly random distributed over $\{0,1\}^t$, x is uniformly distributed over $\{0,1\}^l$, then there is a circuit $C' : \{0,1\}^l \to \{0,1\}^t$ of size $O(gt^2/\epsilon^2)$ such that

$$\Pr_{x}[C'(x) = f(x)] \geq \epsilon^3/(500t).$$

Proof. As

$$\Pr_{x,r}[C(x, r) = \langle f(x), r \rangle] \geq 1/2 + \epsilon,$$

for at least $\epsilon/2$ fraction of all $x \in \{0,1\}^l$,

$$\Pr_{r}[C(x, r) = \langle f(x), r \rangle] \geq 1/2 + \epsilon/2.$$

Let $T = \{x : \Pr_r[C(x, r) = \langle f(x), r \rangle] \geq 1/2 + \epsilon/2\}$. We have $|T|/2^l \geq \epsilon/2$.
For every x in T, consider the following Goldreich-Levin algorithm GL.

Algorithm A.1: $GL(x)$

Input: $x \in \{0,1\}^l$
Let $L_x = \emptyset$
Generate uniform random strings $r_0, r_1, \ldots, r_{k-1}$ over $\{0,1\}^t$, where $k = \lceil \log(100t/\epsilon^2 + 1) \rceil$.
For every $S \subseteq [k], S \neq \emptyset$, let $r_S = \sum_{j \in S} r_j$.
Let $R = \{r_S \in \{0,1\}^t : \emptyset \neq S \subseteq [k]\}$
for $b_0, b_1, \ldots, b_{k-1} \in \{0,1\}$ **do**
  For every $S \subseteq [k], S \neq \emptyset$, let $b_S = \sum_{j \in S} b_j$
  Let $B = \{b_S \in \{0,1\} : \emptyset \neq S \subseteq [k]\}$
  for $i = 0$ *to* $t-1$ **do**
    $y_i = \mathsf{maj}_{\emptyset \neq S \subseteq [k]}\{C(x, r_S \oplus e_i) \oplus b_S\}$
  end
  Add y to L_x
end
return L_x

We claim that $\forall x \in T$, with probability at least 0.99 over the random variables used in GL, $x \in L_x$. The reason is as follows.

In the algorithm GL, we try all the possibilities of $b_0, \ldots, b_{k-1}$. Consider one special choice of them, saying $b_j = \langle f(x), r_j \rangle, \forall j \in [k]$. As a result, $b_S = \langle f(x), r_S \rangle, \forall S \subseteq [k], S \neq \emptyset$. Now we fix an $x \in T$ and an $i \in [t]$. If $C(x, r_S \oplus e_i) = \langle f(x), r_S \oplus e_i \rangle$, then $C(x, r_S \oplus e_i) \oplus b_S = f(x)_i$. We know that, as $x \in T$,

$$\Pr_{r_S}[C(x, r_S \oplus e_i) = \langle f(x), r_S \oplus e_i \rangle] = 1/2 + \epsilon'/2$$

for $\epsilon' \in [\epsilon, 1]$. So

$$\Pr_{r_S}[C(x, r_S \oplus e_i) \oplus b_S = f(x)_i] = 1/2 + \epsilon'/2.$$

According to the construction of r_S in our algorithm, r_S's, $\forall S \subseteq [k], S \neq \emptyset$ are pairwise independent. Let I_S denote the indicator such that $I_S = 1$ if $C(x, r_S \oplus e_i) \oplus b_S = f(x)_i$ and $I_S = 0$ otherwise. Let $I = \sum_{\emptyset \neq S \subseteq [k]} I_S$. Thus $\mathbf{E} I_S = 1/2 + \epsilon'/2$. So $\mathbf{E} I = (1/2 + \epsilon'/2)(2^k - 1)$. Also, the variance of I_S is $\mathbf{Var}(I_S) = \mathbf{E} I_S^2 - (\mathbf{E} I_S)^2 = \mathbf{E} I_S - (\mathbf{E} I_S)^2 \leq 1/4$. So $\mathbf{Var}(I) = \sum_{\emptyset \neq S \subseteq [k]} \mathbf{Var}(I_S) \leq \sum_{\emptyset \neq S \subseteq [k]} \frac{1}{4} = (2^k - 1)/4$. So according to the Chebyshev's inequality,

$$\Pr[I \leq 1/2(2^k - 1)] \leq \Pr[|I - \mathbf{E} I| \geq \frac{\epsilon'}{2}(2^k - 1)] \leq \frac{\mathbf{Var}(I)}{(\frac{\epsilon'}{2}(2^k - 1))^2} = \frac{1}{\epsilon'^2(2^k - 1)} \leq 1/(100t),$$

as $2^k - 1 \geq 100t/\epsilon^2$ and $\epsilon' \geq \epsilon$. Thus $\Pr[y_i \neq f(x)_i] \leq \Pr[I \leq 1/2(2^k - 1)] \leq 1/(100t)$. According to the union bound, $\Pr[y = f(x)] \geq 1 - t/(100t) = 0.99$.

We know that as we have guessed all the possible values of $b_0, b_1, \ldots, b_{k-1}$, one of them should be correct. So for every $x \in T$, with probability at least 0.99 over $r_0, r_1, \ldots, r_{k-1}$, x is in L_x.

Now we modify this algorithm to construct the circuit C'. Notice that we only need to prove that there exists a circuit C'. So we are going to fix the random variables and the choice of $b_0, b_1, \ldots, b_{k-1}$ in the algorithm.

We first fix those random variables. Assume we use the same random variables $r_0, r_1, \ldots, r_{k-1}$ to get L_x for every $x \in T$. For every $x \in T$, let I_x denote the event that $x \in L_x$. We know that $\mathbf{E} I_x \geq 0.99$. So $\mathbf{E} \sum_{x \in T} I_x \geq 0.99|T|$. So there exists $r'_0, r'_1, \ldots, r'_{k-1}$ such that

$$(\sum_{x \in T} I_x)|_{r_i = r'_i, \forall i \in [k]} \geq 0.99|T|.$$

Denote the event $r_i = r'_i, \forall i \in [k]$ as φ.

After we fix these random variables, we fix the choice of $b_0, b_1, \ldots, b_{k-1}$. We know that for at least $0.99|T|$ number of $x \in T$, $x \in L_x|_\varphi$. For every x, let $y_{x, b_0, \ldots, b_{k-1}}$ denote the element in $L_x|_\varphi$ corresponding to $b_0, b_1, \ldots, b_{k-1}$. As the size of $L_x|_\varphi$ is $2^k \leq 200t/\epsilon^2 + 2$, there exists $b'_0, b'_1, \ldots, b'_{k-1} \in \{0, 1\}$ such that for at least $\frac{0.99|T|}{2^l 2^k} \geq \epsilon^3/(500t)$ fraction of $x \in \{0, 1\}^l$, $y_{x, b'_0, b'_1, \ldots, b'_{k-1}} = f(x)$.

By fixing the random variables and $b_0, b_1, \ldots, b_{k-1}$, according to our algorithm, we have a circuit C', such that for every $i \in [t]$,

$$C'(x)_i = \mathsf{maj}_{\emptyset \neq S \subseteq [k]}\{C(x, r'_S \oplus e_i) \oplus b'_S\}$$

where $\forall S \subseteq [k], S \neq \emptyset$, $r'_S = \sum_{j \in S} r'_j$, $b'_S = \sum_{j \in S} b'_j$.

By Theorem A.2, we know the circuit size for computing majority function over $2^k - 1$ input bits is $O(2^k)$.

As all the random variables and $b_0, \ldots, b_{k-1}$ are fixed, for every $i \in [l]$ and every $S \subseteq [k], S \neq \emptyset$, $r'_S \oplus e_i$ and b'_S are fixed (given bits, not need to be computed by the circuit).

So we can see that for every $i \in [t]$, the circuit size of $C'(\cdot)_i$ is $O(g \cdot 2^k + 2^k) = O(gt/\epsilon^2 + t/\epsilon^2) = O(gt/\epsilon^2)$. So the circuit size of C' is $O(gt^2/\epsilon^2)$.

□

Lemma A.4 ([25] Section 5.3)**.** *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 $\tilde{f} : \{0,1\}^{\tilde{l}=\Theta(l)} \to \{0,1\}^l$ that is $g^{-1/4}$-hard for circuit size $g^{1/4}$.*

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

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 $|v_1| = l, |w| = \Theta(l)$. So $\tilde{l} = |a| + |v_1| + |w| = \Theta(l)$.

Now we prove Lemma A.1.

Proof. By Lemma A.4, 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 $\tilde{f} : \{0,1\}^{\tilde{l}=\Theta(l)} \to \{0,1\}^l$ that is $\tilde{\epsilon} = g^{-1/4}$-hard for circuit size $\tilde{g} = g^{1/4}$.

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

Now consider $f' : \{0,1\}^{\tilde{l}+l} \to \{0,1\}$ which is as follows.

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

where $|s| = l$.

By Theorem A.3, f' is $(1/2 - \epsilon)$-hard for circuits of size $g' = \Theta(\tilde{g}\epsilon^2 l^{-2}) = \Theta(g^{1/4}\epsilon^2 l^{-2})$ where ϵ can be such that $\epsilon^3/(500l) \geq \tilde{\epsilon}$. That is $\epsilon \geq (500l)^{1/3} g^{-1/12}$.

□

ECCC ISSN 1433-8092
http://eccc.hpi-web.de