Randomness-Optimal Oblivious Sampling[*]

David Zuckerman[†]

December 12, 1997

Abstract

We present the first efficient oblivious sampler that uses an optimal number of random bits, up to an arbitrary constant factor bigger than 1. Specifically, for any $\alpha > 0$, it uses $(1+\alpha)(m+\log\gamma^{-1})$ random bits to output $d = \mathrm{poly}(\epsilon^{-1}, \log\gamma^{-1}, \mathrm{m})$ sample points $z_1, \ldots, z_d \in \{0,1\}^m$ such that for any function $f : \{0,1\}^m \to [0,1]$,

$$\mathbf{Pr}\left[\left| \frac{1}{d}\sum_{i=1}^{d} f(z_i) - \mathbf{E}f \right| \le \epsilon \right] \ge 1-\gamma.$$

Our proof is based on an improved extractor construction. An extractor is a procedure which takes as input the output of a defective random source and a small number of truly random bits, and outputs a nearly-random string. We present the first optimal extractor, up to constant factors, for defective random sources with constant entropy rate.

We give applications to constructive leader election and reducing randomness in interactive proofs.

1 Introduction

Randomization has proved extremely useful in almost all areas of computer science, such as algorithms, Monte Carlo simulations, cryptography, distributed computing, and network constructions. These uses of randomness have a drawback, however: computers typically don't have access to many truly random bits.

Researchers have taken two different approaches to deal with this problem. One approach is to try to minimize the number of random bits required for a particular task. If the number of random bits can be sufficiently reduced (usually this means to $O(\log n)$), then a deterministic algorithm can often be created by cycling through all possible choices for the random seed (see e.g. [Lub86]). The other approach is to assume that the only random sources available are defective, or weak (e.g. [Blu86, SV86]). In other words, the "random strings" they output are not uniformly random, but rather just somewhat random.

In this paper we deal with both approaches. In fact, we show that the two are related: random sampling using few random bits is essentially equivalent to extracting randomness from general

[*]A preliminary version of this paper, entitled "Randomness-Optimal Sampling, Extractors, and Constructive Leader Election," appeared in *Proceedings of the 27th ACM Symposium on Theory of Computing*, pages 286-295, 1996.

[†]Dept. of Computer Sciences, The University of Texas at Austin, Austin, TX 78712, `diz@cs.utexas.edu`. Supported in part by NSF NYI Grant No. CCR-9457799, a David and Lucile Packard Fellowship for Science and Engineering, and an Alfred P. Sloan Research Fellowship.

weak random sources. Moreover, it often helps to think about one approach even if a result is desired in the other. Here, we obtain good samplers by constructing good extractors for weak sources.

Oblivious Samplers

One of the most basic uses of randomness is sampling. Suppose $f : \{0,1\}^m \to [0,1]$ is arbitrary, and we wish to estimate its mean $\mathbf{E}f = 2^{-m}\sum_{z\in\{0,1\}^m} f(z)$. If we know nothing about the function, the most natural approach is to sample randomly. By choosing $d = O(\epsilon^{-2}\log\gamma^{-1})$ sample points $z_1,\dots,z_d$ uniformly and independently, we obtain an estimate $Z = \frac{1}{d}\sum_{i=1}^{d} f(z_i)$ such that $\mathbf{Pr}[|Z - \mathbf{E}f| \geq \epsilon] \leq \gamma$.

We can define more general samplers:

Definition 1.1 *An* (n,m,d,γ,ϵ)-sampler[1] *is a set of two deterministic algorithms* A *and* B*, which works as follows. Let* $f : \{0,1\}^m \to [0,1]$ *be arbitrary. On input an* n*-bit string,* A *outputs* d sample points $z_1,\dots,z_d \in \{0,1\}^m$*. On input* $f(z_1),\dots,f(z_d)$*,* B *produces an estimate* $\hat{\mu}$ *for* $\mathbf{E}f$*. When the* n*-bit string is chosen uniformly at random, the sampler has the property that* $|\hat{\mu} - \mathbf{E}f| \leq \epsilon$ *with probability* $\geq 1-\gamma$*.*

Probably the simplest sampler is one where B simply computes the average of $f(z_1),\dots,f(z_d)$. Such a sampler is called oblivious:

Definition 1.2 *An* (n,m,d,γ,ϵ)-oblivious sampler *is a deterministic algorithm which, on input a uniformly random* n*-bit string, outputs a sequence of* d sample points $z_1,\dots,z_d \in \{0,1\}^m$ *such that for* any *function* $f : \{0,1\}^m \to [0,1]$*, we have* $|\frac{1}{d}\sum_{i=1}^{d} f(z_i) - \mathbf{E}f| \leq \epsilon$ *with probability* $\geq 1-\gamma$*.*

Not surprisingly, many researchers have worked on producing efficient samplers using few random bits. A summary is given below. For us, the interesting ranges of parameters are $\epsilon = 1/\mathrm{poly(m)}$ and $\gamma = \exp(-\Theta(m))$.

Due to	Method	Obliv?	Random Bits	Sample Points
[CEG95]	Lower Bound	Any	$m + \log\gamma^{-1} - \log(O(d))$	$\Omega(\epsilon^{-2}\log\gamma^{-1})$
Standard	Full Independence	Yes	$O(m\epsilon^{-2}\log\gamma^{-1})$	$O(\epsilon^{-2}\log\gamma^{-1})$
[CG89]	Pairwise Independence	Yes	$2m$	$O(\epsilon^{-2}\gamma^{-1})$
[Gil93]	Random Walks on Expanders	Yes	$m + O(\epsilon^{-2}\log\gamma^{-1})$	$O(\epsilon^{-2}\log\gamma^{-1})$
[BGG93]	Pair. Ind. + RW's on Expan.	No	$2m + O(\log\gamma^{-1})$	$O(\epsilon^{-2}\log\gamma^{-1})$
[GW97]	[BGG93] + Expan./Hashing	No	$m + O(\log\gamma^{-1})$	$O(\epsilon^{-2}\log\gamma^{-1})$
[BR94]	Iterated Sampling	Yes	$O(m + (\log m)\log\gamma^{-1})$	$\mathrm{poly}(\epsilon^{-1}, \log\gamma^{-1}, \log \mathrm{m})$
Here	Hash-Based Extractors	Yes	$(1+\alpha)(m + \log\gamma^{-1})$ $\alpha > 0$ any constant	$\mathrm{poly}(\epsilon^{-1}, \log\gamma^{-1}, \mathrm{m})$

Note that the pairwise independent sampler doesn't even run in polynomial time for the parameters of interest. Thus the only other sampler using a constant times the optimal number of random bits is that of [BGG93] (and that of [GW97] building upon it). That sampler also has

[1] Note that in this definition and future ones there are 5 parameters. It may help to remember that the first three refer to lengths of inputs and outputs: the first is the length of the input, the second is the length of one output sample, and the third is the number of samples. The last two paramenters refer to the quality of the sampler.

the advantage of using an optimal number of sample points, up to a constant factor, whereas ours uses a larger but polynomial number. On the other hand, our sampler has the advantages of being oblivious and using an optimal number of random bits up to an arbitrary constant factor bigger than 1. The previous best oblivious sampler used $O(\log m)$ random bits times the optimal [BR94]. The importance of obliviousness may not be obvious, but it is what is needed for the applications in this paper and in [BR94]. Our sampler also runs in NC.

A corollary of our sampler construction is an NC algorithm for deterministic amplification. In particular, our sampler can be used to convert a BPP (or BPNC) algorithm using m random bits and achieving error $1/3$ into one using $(1+\alpha)(m+k)$ random bits and achieving error 2^{-k}, for any $\alpha > 0$. The only known previous technique requires a larger constant times $m+k$ even to amplify RP. It is based on random walks on expanders [AKS87, CW89, IZ89] (the sampler of [BGG93] builds on this). However, that uses $O(k)$ samples, whereas ours uses poly(m, k) samples. Using the nice structure of the expanders of [GG81], that can also be made to run in NC.

Non-Constructive Upper Bound

Note that for small γ, our oblivious sampler uses fewer random bits than the non-explicit construction given in [CEG95], where $m + 2\log\gamma^{-1} + \log\log(2\epsilon)^{-1}$ random bits are used. Here we rectify this situation by showing non-constructively that there is a sampler taking d samples and using $m + \log\gamma^{-1} - \log(\epsilon^2 d) + 4$ random bits, for $d \geq 2(\log\gamma^{-1})/\epsilon^2$. We achieve a better bound than [CEG95] by viewing the sampler as an extractor.

Extractors for Weak Random Sources

Our oblivious samplers are constructed by viewing them in the essentially equivalent form of extractors for general weak random sources. This view turns out to be quite helpful. Besides giving our main construction and the non-explicit construction above, it can be used to prove an unintuitive proposition by translating a result in [WZ95] into the language of samplers. In particular, if there is an efficient sampler that uses a constant times optimal number of random bits, then there is one using the optimal number times an arbitrary constant factor bigger than one. The only loss is that the new sampler uses a polynomially larger number of samples.

First we explain what we mean by general weak random sources. Many models of weak random sources have been studied (e.g. [Blu86, SV86, CG88, CFG$^+$85, CW89]). We associate a source with the probability distribution by which it outputs a string. Extending models in [CG88, SV86], the most general model was studied in [Zuc90]:

Definition 1.3 *A distribution D on $\{0,1\}^n$ is called a δ-*source *if for all $x \in \{0,1\}^n$, $D(x) \leq 2^{-\delta n}$.*

Improving upon [Zuc90], it was shown in [Zuc96] how to simulate BPP using the output from such a source, for δ a constant. In [NZ96], an extractor was defined and constructed:

Definition 1.4 $E : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ *is an* $(n, m, t, \delta, \epsilon)$-extractor *if, for x chosen according to any δ-source on $\{0,1\}^n$ and y chosen uniformly at random from $\{0,1\}^t$, $E(x, y)$ is within statistical distance*[2] *ϵ from the uniform distribution on $\{0,1\}^m$.*

In previous definitions, $E(x, y)\circ y$ was required to be close to uniform. Yet this stronger condition was not used in the important applications ([NZ96, WZ95, Zuc96]; see [Nis96] for a survey). We can achieve the stronger condition; however, our proofs are cleaner without it.

[2]See Section 3.1 for a definition of statistical distance, or variation distance.

Viewing δ, n, and ϵ as given, the goal is to make t as small as possible and m as large as possible. This can yield a simulation of BPP using a δ-source. Namely, given a BPP machine M which uses m random bits to achieve error .1 and an $(n, m, t, \delta/2, 1/3)$-extractor, cycling through all possible t-bit strings yields a simulation of BPP using n bits from a δ-source and requiring 2^t runs of M [Zuc96]. (What is important above is that $.1 + 1/3 < 1/2$; an exponentially small error in the simulation is achieved because the extractor is designed for a $\delta/2$-source rather than a δ-source.) Thus for this purpose it is desirable to have $t = O(\log n)$ and m as large as possible, but at least $n^{\Omega(1)}$.

Most of the recent progress in this area was made for $\delta = o(1)$ [SZ94, SSZ95, Ta-96]. In particular, building upon [SZ94], Ta-Shma [Ta-96] gave an extractor which uses $t = \text{polylog}(n, \epsilon^{-1})$ truly random bits to extract all δn bits of randomness. Although this work is generally more impressive than the case of constant δ, constant δ was enough to achieve most of the applications [NZ96, WZ95, Zuc96]. In this paper, we achieve the optimal results for constant δ, up to constant factors. It should be noted, however, that the optimal t up to constant factors results in optimal 2^t up to polynomial factors. This 2^t is the relevant term for the time of the BPP simulation and the number of samples output by the oblivious sampler.

The following is a history of work for constant δ; the big-Oh terms hide constants related to δ.

Due to	t truly random bits	m output bits
Lower bound [NZ96]	$\Omega(\log n + \log \epsilon^{-1})$	$\lceil \delta n + t \rceil$
[NZ96]	$O((\log n + \log \epsilon^{-1}) \log n)$	$\Omega(n)$
[Zuc96]	$O(\log n + \epsilon^{-2})$	$n^{\Omega(1)}$
[WZ95]	$O((\log n + \log \epsilon^{-1}) \log n)$	$(\delta - \alpha)n$, any $\alpha > 0$
[SZ94]	$O(\log n + \log \epsilon^{-1})$	$\delta^{\log^* n} n$
Here	$O(\log n + \log \epsilon^{-1})$	$(\delta - \alpha)n$, any $\alpha > 0$

Thus, the main technical contribution of this paper is to eliminate the $\delta^{\log^* n}$ factor in the number of output bits of the extractor in [SZ94]. This yields the first simulation of BPP using a linear number of random bits from a δ-source. That is, if the original BPP algorithm uses m random bits to achieve error probability $1/3$, then the simulation will use $O(m)$ bits from a δ-source to achieve error 2^{-m}.

Constructive Leader Election and Dispersers

In the leader election problem, there are M processors or players. All communication is by broadcast and hence public. That is, each player knows the sender and content of each message. The goal is to elect a leader. The difficulty is that there is a coalition of βM players that want one of their members elected and may not follow the protocol. This problem is well-studied in the cryptographic setting, where there is a constant expected time Byzantine agreement protocol [FM97]. Here we study the full information model: the bad coalition can have infinite computational power, and thus cryptographic tools like one-way functions are useless. A protocol is β-immune if regardless of which βM players are dishonest, the protocol chooses an honest leader with probability bounded away from 0. Note that such a protocol can also be used to flip a coin that has probability bounded away from 0 and 1 of being heads: the elected leader simply flips the coin and broadcasts the result.

Alon and Naor [AN93] were the first to exhibit a protocol which is β-immune for some constant $\beta > 0$. Non-constructively, [BN] improved upon [AN93] to show that there exists a β-immune protocol for any constant $\beta < 1/2$, which is best possible [Sak89]. These protocols require a linear number of rounds. The first sublinear protocol was given in [CL95], and recently [ORV94] exhibited an $O(\log M)$ round protocol that is β-immune for β a sufficiently small constant. They also showed non-constructively the existence of an $O(\log M)$ round protocol that is β-immune for any fixed $\beta < 1/2$.

Here we give a constructive $O(\log M)$ round protocol that is β-immune for any fixed $\beta < 1/2$. We also resolve another unsatisfactory feature of the [ORV94] protocol: in that protocol each player had to send polynomially many bits per round. In our protocol the players need only send $\log M$ bits per round. We rely heavily on the work of [ORV94], but modify their protocol to make the analysis even simpler.

For this application, it is helpful to view the extractor graph-theoretically. The natural way to do this yields a family of highly-expanding uneven bipartite graphs, which have been called dispersers [San87, Sip88, CW89]. The dispersers constructed in this way are stronger than those that can be constructed using eigenvalues in the natural way, and have been used to give near-optimal explicit constructions for superconcentrators, nonblocking networks, and algorithms for sorting and selecting in rounds [WZ95]. Applying the dispersers to appropriately pick committees in the [ORV94] protocol gives our constructive result.

We remark that for this application to leader election, we do not need the new extractor; even that in [NZ96] would have sufficed. However, the new extractor makes this construction cleaner.

Interactive Proofs with Few Random Bits

Oblivious samplers were used in [BR94] to reduce the number of random bits needed for interactive proofs. Using a theorem in [BR94], our oblivious sampler implies the following. Given a $2g(n)$ round AM proof for a language L in which Arthur sends $l = l(n)$ random bits per round and Merlin responds with a $q = q(n)$ bit string, we can construct a $g = g(n)$ round AM proof for L in which Arthur sends $O(l + q)$ random bits per round and Merlin's response remains of polynomial length. The previous best bound was $O(l + (q + \log g) \log l)$ random bits [BR94].

We also mention that our extractor has been used by [AW] to give pseudo-random generators for space-bounded machines, yielding a smooth tradeoff between the generators of [Nis92] and [NZ96].

2 Equivalence of Samplers and Extractors

In this section we show the essential equivalence of samplers and extractors. It is helpful to do this by looking graph-theoretically.

2.1 Graph-Theoretic View of Samplers

We now define an uneven bipartite graph that is equivalent to a sampler. To do this, we put the $N = 2^n$ strings representing the random bits on the left, and the $M = 2^m$ strings representing the sample points on the right. We connect a point x on the left with the d sample points on the right that the sampler outputs on input x. Thus in the following definition think of $d \ll M < N$. Note that we use capital letters for sets and for quantities that are usually exponentially large.

Definition 2.1 *An* (N, M, d, K, ϵ)-approximating disperser *is a bipartite multigraph on independent sets* V *and* W*, such that* $|V| = N$*,* $|W| = M$*, the degree of every node in* V *is* d*, and the following holds. Let* $f : W \to [0,1]$ *be arbitrary, and call* $v \in V$ *bad if* $|\frac{1}{d}\sum_{w\in\Gamma(v)} f(w) - \mathbf{E}f| > \epsilon$*. (Here* $\Gamma(v)$ *denotes the neighbors of* v*.) Then the number of bad vertices in* V *is at most* K*.*

We use the name approximating disperser because it is related to two types of dispersers previously defined: one corresponding to RP, and the other to BPP (see [San87, Sip88, CW89]). For the RP type, a vertex is bad if none of its neighbors lie in some fixed subset $S \subset W$, $|S| \geq |W|/2$. For the BPP type, a vertex is bad if a majority of its neighbors lie outside some fixed subset $S \subset W$, $|S| \geq 2|W|/3$. Note that an approximating disperser is stronger than an RP or BPP type disperser. The dispersers we construct, and that were constructed in [Zuc90, Zuc96, NZ96, WZ95, SSZ95], are better than what can be achieved using eigenvalues and have had numerous applications (see [WZ95, Nis96]).

We also need a notion of efficiently constructibility:

Definition 2.2 *An* (N, M, d, K, ϵ)*-approximating disperser is* efficiently constructible *if there is an efficient algorithm which, on input a node* $v \in V$ *and index* $i \in \{1, 2, \ldots, d\}$*, computes the* i^{th} *neighbor of* v*.*

Since our constructions work in NC, we use the word "efficient" above to mean in NC. Note that the size of the input to this algorithm is $\log N$, so the parallel running time should be polynomial in $\log\log N$.

We defined approximating-dispersers so that the following lemma is obvious:

Lemma 2.3 *There is an efficient* $(n, m, d, \gamma, \epsilon)$*-oblivious sampler iff there is an efficiently constructible* $(2^n, 2^m, d, \gamma 2^n, \epsilon)$*-approximating disperser.*

2.2 Equivalence to Extractors

We will discuss the difference between samplers and extractors by looking graph-theoretically. Approximating dispersers, and hence samplers, say that the number of bad vertices is small. Extractors say that a large enough set of vertices is good. These are basically equivalent: a bad set contains many bad vertices, and the set of bad vertices is a bad set. We need to be slightly careful: conceivably bad vertices which overestimate $\mathbf{E}f$ can combine with bad vertices which underestimate $\mathbf{E}f$ to form a good set. We get around this by dividing the bad vertices into those which overestimate and those which underestimate.

There is also another difference. Recall that the output of an extractor is required to be within statistical distance ϵ of the uniform distribution, which we call ϵ-quasi-random:

Definition 2.4 *Let* D_1 *and* D_2 *be two distributions on the same space* S*. The* variation distance *(or* statistical distance*) between them is*

$$\|D_1 - D_2\| = \max_{T\subseteq S} |D_1(Y) - D_2(Y)| = \frac{1}{2}\sum_{s\in s} |D_1(s) - D_2(s)|.$$

Definition 2.5 *A distribution* D *on* S *is called* ϵ-quasi-random *(on* S*) if the distance between* D *and the uniform distribution on* S *is at most* ϵ*.*

Thus, extractors are defined to have error at most ϵ with respect to subsets, which are functions into $\{0,1\}$, whereas samplers are defined with respect to more general functions into $[0,1]$. Yet while it is not clear that samplers good for $\{0,1\}$ functions are also good for $[0,1]$ functions, it is straightforward to see this for extractors.

Lemma 2.6 *Let $E : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ be an (n,m,t,δ,ϵ)-extractor and let $f : \{0,1\}^m \to [0,1]$ be arbitrary. Let x be chosen according to any δ-source on $\{0,1\}^n$ and y be chosen uniformly at random from $\{0,1\}^t$. Then the expected value of f on the output $E(x,y)$ differs from its true expectation $\mathbf{E}f$ by at most ϵ.*

Proof. The main point is that f can be expressed as a convex combination of functions $g_i : \{0,1\}^m \to \{0,1\}$; that is, $f = \sum_i \lambda_i g_i$, $\lambda_i \geq 0$, $\sum_i \lambda_i = 1$. To see this, let $z_1, \ldots, z_{2^m}$ be the points of $\{0,1\}^m$ ordered so that $f(z_1) \leq f(z_2) \leq \ldots \leq f(z_{2^m})$. We can then define $g_i(z_j) = 1$ if $j \geq i$, and 0 otherwise. Letting $\lambda_i = f(z_i) - f(z_{i-1})$, with the convention that $f(z_0) = 0$, we get that $f(z) = \sum_{i=1}^{2^m} \lambda_i g_i(z)$.

Let Z be the random variable equal to the output of the extractor when the inputs are chosen as required by the lemma. Let U be uniformly random on $\{0,1\}^m$. Since E is an extractor, $|\mathbf{E}g_i(Z) - \mathbf{E}g_i(U)| \leq \epsilon$. We then use the triangle inequality:

$$\begin{aligned} |\mathbf{E}f(Z) - \mathbf{E}f(U)| &= |\sum_i \lambda_i(\mathbf{E}g_i(Z) - \mathbf{E}g_i(U))| \\ &\leq \sum_i \lambda_i |\mathbf{E}g_i(Z) - \mathbf{E}g_i(U)| \\ &\leq \sum_i \lambda_i \epsilon = \epsilon. \end{aligned}$$

□

We now show that good extractors are good oblivious samplers:

Proposition 2.7 *If there is an efficient (n,m,t,δ,ϵ)-extractor, then there is an efficiently constructible $(2^n, 2^m, 2^t, 2^{1+\delta n}, \epsilon)$-approximating disperser, and hence an efficient $(n, m, 2^t, 2^{1-(1-\delta)n}, \epsilon)$-oblivious sampler.*

Proof. The disperser is defined on independent sets $V = \{0,1\}^n$ and $W = \{0,1\}^m$. For each $y \in \{0,1\}^t$, $x \in V$ is connected with $E(x,y)$. Let $f : W \to [0,1]$ be arbitrary, and let B be the set of bad vertices (according to Definition 2.1) in V. Assume without loss of generality that the set $B' = \{v \in V : \frac{1}{d}\sum_{w \in \Gamma(v)} f(w) > \mathbf{E}f + \epsilon\}$ is at least half the size of B. Since the weak source which outputs a uniformly random element from B' violates the conclusion of Lemma 2.6 for f, $|B'| < 2^{\delta n}$ and $|B| \leq 2|B'| < 2^{1+\delta n}$. □

Next we show that an approximating disperser yields an extractor:

Proposition 2.8 *If there is an efficiently constructible $(2^n, 2^m, 2^t, 2^{\delta n}, \epsilon)$-approximating disperser, then for any $\delta' > \delta$ there is an efficient $(n, m, t, \delta', \epsilon + 2^{(\delta - \delta')n})$-extractor.*

Proof. Define the extractor E in an analogous way to Proposition 2.7. Suppose D is an arbitrary δ'-source on $\{0,1\}^n$ and S is an arbitrary subset of $\{0,1\}^m$, say of size $p2^m$. By the definition of approximating disperser, less than $2^{\delta n}$ strings in $\{0,1\}^n$ are bad with respect to the indicator function for S, and all good strings provide an estimate of p that is within ϵ. Since each bad string has probability at most $2^{-\delta' n}$ according to D,

$$|\mathbf{Pr}[E(x,y) \in S] - p| \leq 2^{\delta n} \cdot 2^{-\delta' n} + \epsilon,$$

as required. □

This equivalence allows us to show that oblivious samplers good only for $\{0,1\}$ functions (called oblivious $\{0,1\}$-samplers) are also good for general $[0,1]$ functions:

Proposition 2.9 *If there is an efficient $(n, m, d, \gamma, \epsilon)$-oblivious $\{0,1\}$-sampler, then there is an efficient $(n, m, d, 2\gamma/\epsilon, 2\epsilon)$-oblivious sampler.*

Proof. Suppose there is an efficient $(n, m, d, \gamma, \epsilon)$-oblivious $\{0,1\}$-sampler. Then analogously to Lemma 2.3, there is an efficient $(2^n, 2^m, d, \gamma 2^n, \epsilon)$-approximating $\{0,1\}$-disperser. We set $2^{(\delta-\delta')n} = \epsilon$ and use $2^{\delta n} = \gamma 2^n$ to get $\delta' n = \log(\gamma 2^n/\epsilon)$. Note that the proof of Proposition 2.8 only requires an approximating $\{0,1\}$-disperser. This implies that there is an efficient $(n, m, \log d, (\log(\gamma 2^n/\epsilon))/n, 2\epsilon)$ $\{0,1\}$-extractor. By Lemma 2.6, a $\{0,1\}$-extractor is an extractor with the same parameters. Thus, by Proposition 2.7, there is an efficient $(n, m, d, 2\gamma/\epsilon, 2\epsilon)$-oblivious sampler. □

2.3 Strong Versions

It is sometimes useful to have a somewhat stronger form of oblivious sampler, which corresponds to a stronger form of approximating disperser and extractor.

Definition 2.10 *A* strong $(n, m, d, \gamma, \epsilon)$-oblivious sampler *is a deterministic algorithm which on input a uniformly random n-bit string outputs a sequence of points $z_1, \ldots, z_d \in \{0,1\}^m$ such that: for any collection of d functions $f_1, \ldots, f_d : \{0,1\}^m \to [0,1]$,*

$$\mathbf{Pr}\left[\left|\tfrac{1}{d}\textstyle\sum_{i=1}^{d}(f_i(z_i) - \mathbf{E}f_i)\right| \leq \epsilon\right] \geq 1 - \gamma.$$

Definition 2.11 *A* strong (N, M, d, K, ϵ)-approximating disperser *is a bipartite multigraph on independent sets V and W, such that $|V| = N$, $|W| = M$, every node $v \in V$ has d ordered neighbors $\Gamma_1(v), \ldots, \Gamma_d(v)$, and the following holds. Let $f_i : W \to [0,1]$, $i = 1, \ldots, d$ be arbitrary, and call $v \in V$ bad if $|\frac{1}{d}\sum_{i=1}^{d}(f_i(\Gamma_i(v) - \mathbf{E}f_i)| > \epsilon$. Then the number of bad vertices in V is at most K.*

Analogous to Lemma 2.3, we have:

Lemma 2.12 *There is an efficient strong $(n, m, d, \gamma, \epsilon)$-oblivious sampler iff there is an efficiently constructible strong $(2^n, 2^m, d, \gamma 2^n, \epsilon)$-approximating disperser.*

We can also define a strong form of extractor (which is the way it was originally defined), which is nearly equivalent to a strong oblivious sampler:

Definition 2.13 *$E : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ is a strong $(n, m, t, \delta, \epsilon)$*-extractor *if, for x chosen according to any δ-source on $\{0,1\}^n$ and y chosen uniformly at random from $\{0,1\}^t$, $E(x,y) \circ y$ is ϵ-quasi-random (where $\circ$ denotes concatenation).*

A similar proof to Lemma 2.6 yields

Lemma 2.14 *Let $E : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ be a strong (n,m,t,δ,ϵ)-extractor and let $f : \{0,1\}^m \times \{0,1\}^t \to [0,1]$ be arbitrary. Let x be chosen according to any δ-source on $\{0,1\}^n$ and y be chosen uniformly at random from $\{0,1\}^t$. Then the expected value of f on $E(x,y) \circ y$ differs from its true expectation $\mathbf{E}f$ by at most ϵ.*

Similar proofs to Propositions 2.7 and 2.8 yield the following corresponding propositions:

Proposition 2.15 *If there is an efficient strong (n,m,t,δ,ϵ)-extractor, then there is an efficiently constructible strong $(2^n, 2^m, 2^t, 2^{1+\delta n}, \epsilon)$-approximating disperser, and hence an efficient strong $(n, m, 2^t, 2^{1-(1-\delta)n}, \epsilon)$-oblivious sampler.*

Proof. The disperser is defined on independent sets $V = \{0,1\}^n$ and $W = \{0,1\}^m$. For each $y \in \{0,1\}^t$, $x \in V$ is connected with $E(x,y)$. Let $f_i : W \to [0,1]$, $i = 1, \ldots, d$, be arbitrary, and let $f(x,i) = f_i(x)$, so $\mathbf{E}f = \frac{1}{d}\sum_{i=1}^{d} \mathbf{E}f_i$. Let B be the set of bad vertices (according to Definition 2.11) in V. Assume without loss of generality that the set $B' = \{v \in V : \frac{1}{d}\sum_{i=1}^{d} f_i(\Gamma_i(v) > \mathbf{E}f + \epsilon\}$ is at least half the size of B. If x is chosen according to a δ-source which outputs a uniformly random element from B' and i is chosen uniformly at random, then the expectation of f is the average of quantities greater than $\mathbf{E}f + \epsilon$, and hence is greater than $\mathbf{E}f + \epsilon$. Therefore, by Lemma 2.14, $|B'| < 2^{\delta n}$ and $|B| \leq 2|B'| < 2^{1+\delta n}$. □

Proposition 2.16 *If there is an efficiently constructible strong $(2^n, 2^m, 2^t, 2^{\delta n}, \epsilon)$-approximating disperser, then for any $\delta' > \delta$ there is an efficient strong $(n, m, t, \delta', \epsilon + 2^{(\delta-\delta')n})$-extractor.*

Proof. Similar to Proposition 2.8. □

2.4 Extractor View Helps

We have already seen one example where the extractor view of samplers helped: that samplers good for $\{0,1\}$-functions are also (almost as) good for $[0,1]$-functions. Here we give an example of how a surprising proposition about samplers becomes more intuitive when viewed from the point of view of extractors. The proposition about samplers says that if there is an efficient sampler using a constant times the optimal number of random bits, then there is one using the optimal number times an arbitrary constant factor bigger than one. The price we pay is a larger polynomial for the number of samples:

Proposition 2.17 *If there is an efficient $(O(m + \log \gamma^{-1}), m, d, \gamma, \epsilon)$-oblivious sampler, then for any $\alpha > 0$ there is an efficient $((1+\alpha)(m + \log \gamma^{-1}), m, d^{O(1)}, \gamma, \epsilon)$-oblivious sampler.*

Note that in such a statement, and in similar statements throughout the paper, we refer to one sampler, but we really mean a family of samplers.

The corresponding proposition about extractors is:

Proposition 2.18 *[WZ95] If there is an efficient $(n, \Omega(\delta n), t, \delta, \epsilon)$-extractor E, then for any $\alpha > 0$ there is an efficient $(n, (\delta - \alpha)n, O(t), \delta, \epsilon)$-extractor E'.*

The intuition for the proof, given in [WZ95], is as follows. When we extract randomness from x using E and truly random bits y_1, even conditional on the output bits $E(x, y_1)$ there is a lot of randomness left in x. There were δn bits of randomness to begin with, and the length of $E(x,y)$ is only βn for some $\beta < \delta$, so there should be $(\delta - \beta)n$ bits of randomness left. We can extract the remaining randomness by using E again with independent truly random bits y_2. This time E is designed to extract from a $(\delta - \beta)$-source. Continuing in this manner, we get $E'(x, y_1 \circ \ldots \circ y_a) = E(x, y_1) \circ \ldots \circ E(x, y_a)$ for an appropriate constant a, where $\circ$ denotes concatenation. The proof shows that this intuition is close enough to the truth.

When we translate this proof to the language of samplers, we see that we build a new sampler by increasing the length of the sample points, rather than decreasing the number of random bits. Thus, we are exploiting the fact that we have a family of samplers, and not just one sampler.

In fact, we exploit this in the extractor proof above as well, although to a lesser extent. The extractor E' will have error $O(\epsilon)$, but by starting with an extractor E with error a small enough constant fraction of ϵ, we can make E' have error ϵ.

2.5 A Non-Explicit Construction

Another instance where the extractor view helps is in a non-explicit construction. We show that there exists samplers with better parameters than achieved in [CEG95] by first showing the existence of the corresponding extractor.

Proposition 2.19 *Let positive n, m, δ and ϵ be given, and set $k = \delta n$. Suppose $d \geq (2 \ln 2)(2^{m-k} + n - k + 3)/\epsilon^2$. Then there is an $(n, m, t = \log d, \delta, \epsilon)$-extractor.*

Note that we can take $k = m$, in which case $d = O(n/\epsilon^2)$ and $t = \log n + 2 \log \epsilon^{-1} + O(1)$.

Proof. We show that picking the outputs of the extractor uniformly at random from $\{0,1\}^t$ yields an extractor of the above quality with high probability. It may be easier to visualize the extractor as a bipartite graph on $\{0,1\}^n \times \{0,1\}^m$ in the natural way. For each vertex in $\{0,1\}^n$, we pick d neighbors from $\{0,1\}^m$ uniformly at random.

Note that since any δ-source is a convex combination of flat δ-sources, where a flat δ-source outputs a uniformly random element from a set of size 2^k, it suffices to show the extractor property for flat sources. Thus, the random graph above is not an extractor if and only if there exists a set $B \subseteq \{0,1\}^n$ of size 2^k and a set $S \subseteq \{0,1\}^m$ such that the following holds. If x is chosen uniformly at random from B, and z is a uniformly random neighbor of x, then $|\mathbf{Pr}[z \in S] - |S|2^{-m}| > \epsilon$. The probability q that such B and S exists can be upper bounded by summing over all B and S:

$$q \leq \binom{2^n}{2^k} 2^{2^m} (2e^{-\epsilon^2 2^k d/2}).$$

Now

$$\binom{2^n}{2^k} \leq 2^{2^n H(2^{k-n})} \leq 2^{(n-k+2)2^k},$$

where we use $H(p) \leq p \log(4/p)$, where H is the binary entropy function. This gives

$$q \leq 2^{1+2^m+2^k(n-k+2-(\log e)\epsilon^2 d/2)}.$$

By the bound on d,

$$n - k + 2 - (\log e)\epsilon^2 d/2 \leq -(2^{m-k} + 1).$$

Thus

$$q \leq 2^{1+2^m-2^k(2^{m-k}+1)} < 1,$$

as required. □

This yields the following sampler:

Proposition 2.20 *Let $0 < \epsilon, \gamma < 1$ and integers $d, m \geq 2$ be given, such that $d_0 = d - 2(\ln 2)(4 + \log \gamma^{-1})/\epsilon^2$ is positive. Let $n \geq m + (\log \gamma^{-1}) + 2 - \log(\epsilon^2 d_0)$. Then there exists an $(n, m, d, \gamma, \epsilon)$ oblivious sampler.*

Note that for $d \geq 2(\log \gamma^{-1})/\epsilon^2$ and larger than some constant, $d_0 \geq d/4$ and we can take the number of random bits to be $n = m + (\log \gamma^{-1}) + 4 - \log(\epsilon^2 d)$.

Proof. We use Proposition 2.7. By using $k = n - 1 - \log \gamma^{-1}$, we get that $d \geq (2 \ln 2)(2^{m-k} + n - k + 3)/\epsilon^2$ is equivalent to $d_0 \geq 2^{m+2-n}(\ln 2)/(\gamma\epsilon^2)$, which in turn is equivalent to the bound on n in the statement of the proposition. Thus Proposition 2.19 gives the proposition. □

3 The Extractor and Sampler Construction

As mentioned earlier, we construct a good sampler by constructing a good extractor, and then using Proposition 2.7. In fact, our extractor is a strong extractor, and we can use Proposition 2.15 to get a strong oblivious sampler.

This section is the most technical. The reader willing to accept Theorems 3.14, 3.16, and 3.17 should skip to the next section.

3.1 Preliminaries

To ease readability we assume, when necessary, that various quantities are integers. It is not difficult to see that this does not affect our analysis. The notation log denotes the logarithm to the base 2.

A convenient fact to remember is that distance between distributions cannot be created out of nowhere. In particular if $f : S \to T$ is any function and D_1, D_2 are distributions on S then $\|f(D_1) - f(D_2)\| \leq \|D_1 - D_2\|$. Here $f(D)$ denotes the distribution of $f(X)$, where X has distribution D. Also if E_1 and E_2 are distributions on Y then $\|D_1 \times E_1 - D_2 \times E_2\| \leq \|D_1 - D_2\| + \|E_1 - E_2\|$.

Definition 3.1 *A distribution D on the space $\{0,1\}^{l_1} \times \{0,1\}^{l_2} \times \cdots \times \{0,1\}^{l_k}$ is called a* block-wise δ-source *if, for $1 \leq i \leq k$ and for all values $x_1 \in \{0,1\}^{l_1}, \ldots, x_i \in \{0,1\}^{l_i}$,*

$$\mathbf{Pr}[X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \leq 2^{-\delta l_i},$$

where the vector of random variables $X_1 \ldots X_k$ is chosen according to distribution D. A block-wise δ-source is the same as the PRB-source of [CG88] except that here the block length is allowed to vary.

3.2 The Construction

In this section, we use the word "efficient" to mean in NC. The reader is advised to ignore the lower bounds on δ and ϵ on the first reading, and think of δ as a constant and $\epsilon = 1/\text{poly(n)}$.

Our construction uses and improves the construction in [SZ94]. The following is from the final version of [SZ94]:

Theorem 3.2 *[SZ94] There is a constant c such that for any $\beta > 0$ and any parameters $\delta = \delta(n)$ and $\epsilon = \epsilon(n)$ with $\delta \geq n^{-1/2 \log^* n}$ and $\epsilon \geq \exp(-\delta^{\log^* n} n^{1-\beta})$, there is an $(n, m = \delta^{\log^* n} n, t = c(\log n + \log \epsilon^{-1}), \delta, \epsilon)$-extractor which runs in NC.*

Remark 3.3 *We can relax the condition on ϵ slightly by using random walks on expanders: we may then take $\epsilon \geq \exp(-\delta^{2\log^* n} n)$. However, this is usually not in the range of interest.*

Thus, our goal is to improve the output length m from $\delta^{\log^* n}$ to $(\delta - \alpha)n$ while not increasing t by more than a constant factor. In fact, by using Proposition 2.18, the tool used in [WZ95], it suffices to improve the output length to $\Omega(n)$. Before we outline our construction, we set up the suitable framework, and outline the construction in [NZ96], which uses ideas from [Zuc90, Zuc96]. As in [NZ96], our extractor is composed of a *block-wise converter* and a *block-wise extractor*, defined explicitly in [SZ94].

Definition 3.4 $E : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^{l_1+\cdots+l_s}$ *is an* $(n, (l_1, \ldots, l_s), t, \delta, \delta', \epsilon)$ block-wise converter *if, for x chosen from a δ-source on $\{0,1\}^n$ and y uniformly from $\{0,1\}^t$, $E(x, y)$ is within ϵ of some block-wise δ'-source with block lengths $l_1, \ldots, l_s$.*

Definition 3.5 $E : \{0,1\}^{l_1+\cdots+l_s} \times \{0,1\}^t \to \{0,1\}^m$ *is an* $((l_1, \ldots, l_s), m, t, \delta, \epsilon)$ block-wise extractor *if, for x chosen from a block-wise δ-source with block lengths $l_1, l_2, \ldots, l_s$ and y uniformly from $\{0,1\}^t$, $E(x, y)$ is ϵ-quasi-random on $\{0,1\}^m$.*

The following trivial lemma, implicit in [NZ96] and explicit in [SZ94], states that together these constitute an extractor:

Lemma 3.6 *Given an efficient $(n, (l_1, \ldots, l_s), t_1, \delta, \delta', \epsilon_1)$ block-wise converter and an efficient $((l_1, \ldots, l_s), m, t_2, \delta', \epsilon_2)$ block-wise extractor, we can construct an efficient $(n, m, t_1 + t_2, \delta, \epsilon_1 + \epsilon_2)$-extractor.*

We always use the same block-wise converter, but recursively build stronger block-wise extractors to get our new extractor. Our block-wise converter is from [NZ96], which uses k-wise independence. Actually, the parameters here come from the improved analysis in the final version of [SZ94] (or the algorithm in [NZ96] using random walks on expanders).

Lemma 3.7 *[NZ96] There is a constant c such that the following holds. Let $\delta \leq 1/2$, n, and n_i, $1 \leq i \leq k$, be such that $\sum_{i=1}^{k} n_i \leq \delta n/4$, and let $\delta' = \delta/c \log \delta^{-1}$. Then for any ϵ there is an efficient $(n, (n_1, n_2, \ldots, n_k), ck(\log n + \log \epsilon^{-1}), \delta, \delta', \epsilon)$ block-wise converter.*

The block-wise extractor in [NZ96] uses a recursive construction of hash functions. It starts with a (weak) extractor built from hash functions: $E(x, h) = h(x) \circ h$. The Leftover Hash Lemma of [ILL89] implies that this is an extractor. A block-wise extractor is then constructed as follows. The block-wise source has blocks with geometrically-decreasing lengths $n_1 > n_2 > \ldots > n_k$. A small number t_k of truly random bits are used to extract m_k bits from the block of length n_k. These m_k bits are then used as the t_{k-1} truly random bits to extract m_{k-1} bits from the block of length n_{k-1}. We continue in this manner until we output m_1 bits from the block of length n_1. The proof in [NZ96] shows that this procedure works (and it makes use of the fact that the procedure starts with the last block and works backwards).

We will use the same recursive construction, except we replace the above weak extractor by an arbitrary extractor. This idea was used in [SZ94], although the following lemma was not stated explicitly, nor exploited in full. The reader should think of $m_i = t_{i-1}$ below; we state it as $m_i \geq t_{i-1}$ as it is sometimes convenient not to worry about making them exactly equal, and then we can ignore the extra bits.

Lemma 3.8 *Let δ, $n_i, t_i, m_i, \epsilon_i$, $1 \le i \le k$, be given such that $m_i \ge t_{i-1}$, $2 \le i \le k$ and there are efficient $(n_i, m_i, t_i, \delta, \epsilon_i)$-extractors E_i, $1 \le i \le k$. Then there is an efficient $((n_1, n_2, \ldots, n_k), m_1, t_k, \delta, \sum_{i=1}^k \epsilon_i)$ block-wise extractor E.*

Proof. Let $(X_1, X_2, \ldots, X_k)$ be the output blocks from a block-wise δ-source with block lengths $n_1, n_2, \ldots, n_k$. Let Y be the truly random input. The extractor E computes $Z_k = Y$ and $Z_{i-1} =$ last t_{i-1} bits of $E_i(X_i, Z_i)$ for $i = k$ down to 1 (we interpret $t_0 = m_1$). We claim that Z_0 is quasi-random to within $\sum_{i=1}^k \epsilon_i$. The proof of this, by induction, is basically the same as the proof in [NZ96] for the special case of this lemma when the E_i correspond to the weak hash extractors described above. We include the proof here for completeness.

We will prove by induction from $i = k$ down to $i = 0$ the following claim, which clearly implies the lemma.

Claim: For any sequence of values $x_1 \ldots x_i$, the distribution of Z_i conditioned on $X_1 = x_1, \ldots, X_i = x_i$, is quasi-random to within $\sum_{j=i+1}^k \epsilon_j$.

This claim is clearly true for $i = k$. Now suppose it is true for i; we prove it for $i-1$. Fix the conditioning $X_1 = x_1, \ldots, X_{i-1} = x_{i-1}$, and let D_i denote the induced distribution on X_i. Since, by the induction hypothesis, for every x_i, the induced distribution on Z_i is quasi-random, we have that the induced distribution on (X_i, Z_i) is within $\sum_{j=i+1}^k \epsilon_j$ of the distribution $D_i \times U_i$, where U_i is the uniform distribution on $\{0,1\}^{t_i}$.

Thus, the distribution of Z_{i-1} is within $\sum_{j=i+1}^k \epsilon_j$ of the distribution obtained by picking x_i according to D_i, and Z_i independently and uniformly at random in $\{0,1\}^{t_i}$. By the definition of extractor, this second distribution is quasi-random to within ϵ_i. The claim follows. □

Using Lemma 3.6 to combine the block-wise converter of Lemma 3.7 and the block-wise extractor of Lemma 3.8 gives an extractor:

Lemma 3.9 *Let n, t, $\delta \le 1/2$, ϵ and $n_i, m_i, t_i, \epsilon_i$, $1 \le i \le k$, be given such that $m_i \ge t_{i-1}$, $2 \le i \le k$ and $2\sum_{i=1}^k \epsilon_i \le \epsilon$. Define δ' as in Lemma 3.7. Further suppose that there are efficient $(n_i, m_i, t_i, \delta', \epsilon_i)$-extractors E_i, $1 \le i \le k$, and $\sum_{i=1}^k n_i \le \delta n/4$. Then there is an efficient $(n, m_1, ck(\log n + \log \epsilon^{-1}) + t_k, \delta, \epsilon)$-extractor, where c is from Lemma 3.7.*

This lemma will be the key to our construction. For the following discussion assume δ is a constant and $\epsilon = 1/\mathrm{poly(n)}$. The idea is to choose the extractors $E_1, \ldots, E_k$ so that $t_k = O(\log n)$, $m_1 = \Omega(n)$, and $k = O(1)$, while ensuring $m_i \ge t_{i-1}$ (the reader can think of $m_i = t_{i-1}$).

A first attempt would be to use the $(n_i, \Omega(n_i), O(\log^2 n_i), \delta, 1/\mathrm{poly(n_i)})$-extractors in [NZ96], with varying n_i, as follows. Setting $n_1 = \delta n/8$, we let E_1 be a $(\delta n/8, m_1 = \Omega(n), t_1 = c\log^2 n, \delta', 1/\mathrm{poly(n)})$-extractor. We now wish to choose E_2 so that $t_2 = O(\log n)$ and $m_2 \ge t_1 = c\log^2 n$. We can do this by choosing n_2 small enough: $n_2 = 2^{\sqrt{\log n}}$. That is, for this n_2 we take E_2 to be an $(n_2, m_2 = \Omega(n_2) \ge t_1, t_2 = O(\log^2 n_2) = O(\log n), \delta', 1/\mathrm{poly(n_2)})$-extractor. Lemma 3.9 then yields an $(n, \Omega(n), O(\log n), \delta, 1/\mathrm{poly(n_2)})$-extractor. This seems to be what we want, except that the error is $1/\mathrm{poly(n_2)}$ rather than $1/\mathrm{poly(n)}$. Indeed, this is enough to get the simulation of BPP using a linear number of bits from a δ-source, for constant δ.

Instead of building E_2 from the original [NZ96] extractor, we use the [SZ94] extractor, given in Theorem 3.2. We can now take $n_2 = n_1 = \delta n/8$. Thus $t_2 = O(\log n)$ and $m_2 = \delta^{\log^* n} n$, which is bigger than $t_1 = O(\log^2 n)$. This gives our result, although in some sense it seems wasteful because $m_2 \gg t_1$. Therefore, for E_1, we use a somewhat more appropriate extractor than the [NZ96] extractor, in order to improve the dependence on δ. This more appropriate extractor is built from the basic [SZ94] extractor given below, which is similar to the [NZ96] extractor except it has much better dependence on δ:

Theorem 3.10 *[SZ94] There is a constant c such that for any $\beta > 0$ and any parameters $\delta = \delta(n)$ and $\epsilon = \epsilon(n)$ with $\delta \leq 1/2$ and $\epsilon \geq \exp(-\delta^2 n^{1-\beta})$, there is an $(n, m = \delta^2 n/c \log \delta^{-1}, t = c \log n(\log n + \log \epsilon^{-1}), \delta, \epsilon)$-extractor which runs in NC.*

We also make use of the following lemma in [WZ95], which is the general, more precise statement corresponding to Proposition 2.18.

Proposition 3.11 *[WZ95] Fix positive integers n and k. Suppose that for each $\delta \in [\eta, 1]$ we are given an efficient $(n, m(\delta), t(\delta), \delta, \epsilon(\delta))$-extractor, where t and ϵ are non-increasing functions of δ. Let $f(\delta) = m(\delta)/(\delta n)$, and suppose f is non-decreasing. Let $a = \ln(\delta/\eta)/f(\eta)$ or, if f grows at least linearly (i.e. $f(c\delta) \geq cf(\delta)$), let $a = 2/f(\eta)$. Then we can construct an efficient $(n, (\delta - \eta)n - k, a \cdot t(\eta), \delta, a(\epsilon(\eta) + 2^{-k}))$-extractor.*

We now choose $\eta = \delta/4$ and $k = \delta n/4$, and apply Proposition 3.11 to Theorem 3.10. This yields an extractor equivalent to the [NZ96] extractor for constant δ, but better than that for $\delta = o(1)$.

Lemma 3.12 *There is a constant c such that for any $\beta > 0$ and any parameters $\delta = \delta(n)$ and $\epsilon = \epsilon(n)$ with $\delta \leq 1/2$ and $\epsilon \geq \exp(-\delta^2 n^{1-\beta})$, there is an $(n, m = \delta n/2, t = c \log n \log \delta^{-1}(\log n + \log \epsilon^{-1})/\delta, \delta, \epsilon)$-extractor which runs in NC.*

This gives:

Lemma 3.13 *There is a constant c such that for $\delta = \delta(n)$ and $\epsilon = \epsilon(n)$ with $n^{-1/2 \log^* n} \leq \delta \leq 1/2$ and $\epsilon \geq \exp(-\delta^{\log^* n} n^{1-\beta})$, there is an $(n, m = \delta^2 n/c \log \delta^{-1}, t = c(\log n + \log \epsilon^{-1}), \delta, \epsilon)$-extractor which runs in NC.*

Proof. Apply Lemma 3.9 with $k = 2$ using the extractors E_1 from Lemma 3.12 and E_2 from Theorem 3.2, with $n_1 = n_2 = \delta n/8$. Note that $t_1 = O(\log n_1 \log(\delta')^{-1}(\log n + \log \epsilon^{-1})/\delta'$ is smaller than $m_2 = (\delta')^{\log^* n_2} n_2$. Thus Lemma 3.9 applies and the output is $m_1 = \delta' n_1/2 = \Theta(\delta^2 n/\log \delta^{-1})$. □

We now use Proposition 3.11 to increase the output length m at the expense of t. In the following theorem we think of α and δ as constants, although we state it more generally.

Theorem 3.14 *There is a constant c such that for any $\beta > 0$ and any parameters $\alpha = \alpha(n) \leq 1/2$, $\delta = \delta(n) \leq 1$, and $\epsilon = \epsilon(n)$, with $n^{-1/(2 \log^* n)} \leq \alpha < \delta$ and $\epsilon \geq \exp(-\alpha^{\log^* n} n^{1-\beta})$, there is an $(n, m = (\delta - \alpha)n, t = c_\alpha(\log n + \log \epsilon^{-1}), \delta, \epsilon)$-extractor which runs in NC, where $c_\alpha = c(\log \alpha^{-1})/\alpha$.*

Proof. Apply Proposition 3.11 to Lemma 3.13 to increase the output length. □

In fact, if we remove part of the output, our extractors are strong extractors. This is because for the block-wise extractors using hash functions, the input hash function h is part of the output. Therefore, if we remove h from the output, then we will have a strong version of block-wise extractor. Furthermore, the block-wise converters and extractors of Lemma 3.7, Theorem 3.2, and Proposition 3.11 hold for strong extractors. We therefore have:

Theorem 3.15 *There is a constant c such that for any $\beta > 0$ and any parameters $\alpha = \alpha(n) \leq 1/2$, $\delta = \delta(n) \leq 1$, and $\epsilon = \epsilon(n)$, with $n^{-1/(2 \log^* n)} \leq \alpha < \delta$ and $\epsilon \geq \exp(-\alpha^{\log^* n} n^{1-\beta})$, there* *is a strong* $(n, m = (\delta - \alpha)n, t = c_\alpha(\log n + \log \epsilon^{-1}), \delta, \epsilon)$*-extractor which runs in NC, where* $c_\alpha = c(\log \alpha^{-1})/\alpha$*.*

This yields a new disperser:

Theorem 3.16 *There is a constant* c *such that for any* $\beta > 0$ *and any parameters* $\delta = \delta(N)$*,* $\epsilon = \epsilon(N)$*, and* $\alpha = \alpha(N)$ *with* $(\log N)^{-1/(2\log^* N)} \le \alpha < \delta \le 1/2$ *and* $\epsilon \ge \exp(-\alpha^{\log^* N}(\log N)^{1-\beta})$*, there is an efficiently constructible strong* $(N, M = N^{\delta-\alpha}, d = ((\log N)/\epsilon)^{c_\alpha}, K = N^\delta, \epsilon)$*-approximating disperser, where* $c_\alpha = c(\log \alpha^{-1})/\alpha$*.*

Proof. Use Proposition 2.15 and Theorem 3.15. Doing this directly would give an extra factor of 2 in front of the N^δ; by changing the definition of δ we can hide this factor of 2 in the constant c.
□

This, in turn, yields a sampler:

Theorem 3.17 *There is a constant* c *such that for any* $\beta > 0$ *and any parameters* $\gamma = \gamma(m)$*,* $\epsilon = \epsilon(m)$*, and* $\alpha = \alpha(m)$ *with* $m^{-1/2\log^* m} \le \alpha \le 1/2$ *and* $\epsilon \ge \exp(-\alpha^{\log^* m} m^{1-\beta})$*, there exists a strong* $(n, m, d, \gamma, \epsilon)$*-oblivious sampler which runs in NC and uses only* $n = (1+\alpha)(m + \log \gamma^{-1})$ *random bits and outputs* $d = ((m + \log \gamma^{-1})/\epsilon)^{c_\alpha}$ *sample points, where* $c_\alpha = c(\log \alpha^{-1})/\alpha$*.*

Proof. Assume α is small enough so that $(1+\alpha)(1-\alpha/2) \ge 1$. Replacing α by $\alpha/2$ and applying Theorem 3.16, there is an efficiently constructible strong $(N, N^{\delta-\alpha/2}, ((\log N)/\epsilon)^{c_\alpha}, N^\delta, \epsilon)$-approximating disperser, where $c_\alpha = c(\log \alpha^{-1})/\alpha$ (note that this c_α is slightly different than the one in Theorem 3.16, because of the change from α to $\alpha/2$). Let $n = (1+\alpha)(m + \log \gamma^{-1})$, $N = 2^n$, and $\delta = 1 - (\log \gamma^{-1})/n$, so $N^\delta = \gamma N$. Then $N^{\delta - \alpha/2} = \gamma N \cdot N^{-\alpha/2} = \gamma(2^m/\gamma)^{(1+\alpha)(1-\alpha/2)} \ge \gamma 2^m$. Now Proposition 2.12 yields the theorem. □

Remark 3.18 *By Remark 3.3, we can relax the lower bound on* ϵ *in the above theorems by using random walks on expanders. We need only assume* $\epsilon \ge \exp(-\alpha^{2\log^* n} n)$ *in Theorems 3.14 and 3.15,* $\epsilon \ge \exp(-\alpha^{2\log^* N}\log N)$ *in Theorem 3.16, and* $\epsilon \ge \exp(-\alpha^{2\log^* m} m)$ *in Theorem 3.17.*

4 Interactive Proofs with Few Random Bits

To obtain our theorem about interactive proofs, we need only the following theorem from [BR94]:

Theorem 4.1 *[BR94] Suppose we have a* $(\mathrm{AM})^{2g}\mathrm{A}$ *(*$g = g(n)$*) proof system for L in which Arthur's messages are of length* $l = l(n)$ *and Merlin's messages are of length* $q = q(n)$*, and suppose we have a strong* $(r = r(n), l, s = s(n), \eta 2^{-q-1}, \eta)$*-oblivious sampler for* $\eta = \eta(n) = 1/6g(n)$*. Then we can construct a* $(\mathrm{AM})^g\mathrm{A}$ *proof system for L in which Arthur uses only* $O(r + \log n)$ *random bits and Merlin's messages are of length* $O(sq)$*.*

As a corollary to Theorem 3.17, we deduce

Theorem 4.2 *Suppose we are given a* $2g(n)$ *round AM proof system for L, where in each round Arthur uses* $l(n)$ *random bits and Merlin responds with a* $q(n)$ *bit string. Then we can construct a* $g(n)$ *round AM proof system for L where in each round Arthur uses only* $O(l(n) + q(n))$ *random bits (and Merlin's response remains of polynomial length).*

5 Constructive Leader Election

Recall that in the leader election problem, there are M players and at most βM dishonest players, $\beta < 1/2$ a constant. All communication is by broadcast and hence public. That is, each player knows the sender and content of each message. The goal is to elect a leader such that an honest player is selected with probability bounded away from 0. The dishonest players may cheat, i.e. not follow protocol and share information, in their attempt to elect a dishonest player.

A protocol proceeds in rounds. In each round a player broadcasts a message. Ideally, the messages would be transmitted in parallel. However, in a distributed system some players may receive messages before others, and dishonest players may wait until they receive other messages before transmitting their own. The players are synchronized between rounds: all messages sent in round i are received before any are sent in round $i + 1$.

As in many leader election protocols, the protocol of [ORV94] works by forming committees. Each committee is a multi-set of players, and each player can be in several committees. One of these committees is selected by some protocol, and then recursively the committee selects a leader from among its own members.

Each recursive level of the [ORV94] protocol is composed of $O(\log M)$ rounds, where in each round some of the committees are eliminated. If there are N' committees left after a round, then in the next round each of the M players eliminates N'/M committees at random (of course, the dishonest players may not eliminate their committees at random). The recursive layer ends the first time there are less than $2M$ committees left (the choice of $2M$ is designed to minimize the impact of round-off errors).[3] Then an arbitrary committee from the remaining ones, say the lowest numbered one, is chosen to recursively select a leader.

We now analyze the behavior of one recursive level. Let $\epsilon < 1/2 - \beta$ be given. Call a committee dangerous if it has more than $\beta + \epsilon$ fraction of dishonest players. The proof of correctness in [ORV94] proceeds by showing that with high probability, a dangerous committee is not selected.

Our modification of the [ORV94] protocol is to select the committees differently. We will use a disperser to select the committees. We also simplify their analysis via the following lemma:

Lemma 5.1 *Let there be M players and N committees. Suppose there is a $p > 0$ such that*

1. *No more than half the committees are eliminated in any round.*
2. *For each round i and remaining committee C, conditional on previous rounds, $\mathbf{Pr}[C \text{ is eliminated in round } i] \geq p$.*
3. *At most $N^{p/2}$ of the committees are dangerous.*
4. *$N \geq (2M)^{2/p}$.*

Then, with probability greater than $1 - 1/M$, the protocol ends after $(\ln N)/p$ rounds and a dangerous committee is not selected.

Proof. We use ideas from [ORV94]. Let $s = \log \frac{N}{2M}$. By the first assumption, the number of committees left after s rounds is at least $2^{-s}N = 2M$, so the protocol has not yet terminated.

[3] Note that our use of M for the number of players and N for the number of committees is the opposite of [ORV94]; however, this will correspond with the letters we use for dispersers.

Note that the first two assumptions of the lemma imply that $p \leq 1/2$. Thus the fourth assumption implies that $N \geq (2M)^4$ and $s \geq \log N^{3/4} = \frac{3}{4}(\log e) \ln N > \ln N$.

From the second assumption of the lemma, the expected number of dangerous committees surviving after s rounds is at most

$$N^{p/2}(1-p)^s < N^{p/2}e^{-ps} < N^{p/2}N^{-p} = N^{-p/2} < 1/(2M).$$

Then, by Markov's inequality, the probability that there exists a dangerous committee after s rounds is less than $1/(2M)$. As the protocol lasts at least s rounds, the probability of the protocol selecting a dangerous committee is less than $1/(2M)$.

Similarly, the expected number of committees left after $s' = (\ln N)/p$ rounds is at most

$$N(1-p)^{s'} < Ne^{-ps'} = 1.$$

Since the protocol ends when there are fewer than $2M$ committees left, the probability that the protocol has not ended after s' rounds is less than $1/(2M)$. Adding the two probabilities gives the lemma. □

We now modify the protocol of [ORV94] to achieve the first two preconditions of the lemma where each player uses at most $\log N$ random bits per round. Number the N' remaining committees $0, 1, 2, \ldots, N'-1$. Each player then picks a uniformly random non-negative integer r less than N', and eliminates committees $r, r+1, r+2, \ldots, r-1+N'/(2M)$, where the committee numbers are viewed modulo N'.

The first condition is easily seen to be satisfied. The probability that a committee does not get eliminated is $(1-1/(2M))^{(1-\beta)M} < e^{-1/4}$, thus satisfying the second condition for $p = 1-e^{-1/4} > 1/5$.

To achieve the third and fourth preconditions of the lemma, we use the $(N, M = N^{p/4}, (\log N)^{O(1)}, N^{p/2}, \epsilon)$-approximating disperser given by Theorem 3.16. The M independent vertices represent the players, and the N independent vertices represent committees. A committee is connected to the players that it contains. By the definition of approximating disperser there are at most $N^{p/2}$ dangerous committees.

This allows us to prove:

Theorem 5.2 *For any constant $\beta < 1/2$ there is an $O(\log M)$ round leader election protocol that is β-immune, where each player transmits only $\log M$ bits per round.*

We remark that there is no need for the players to transmit $\log N = c \log M$ bits per round: they can transmit $\log M$ bits per round if we increase the number of rounds by a factor of c.

Proof. After $s = O(\log M)$ rounds the above protocol reduces the problem to one of size $(\log M)^{O(1)}$. As suggested in [ORV94], we then recurse twice more, and are left with a problem of size $o(\log \log M)$. We can then use the non-constructive sequential protocol of [AN93], proved optimal by [BN], which is constructive and fast enough for such a small problem size. □

We remark that in [ORV94], the committees had size $O(\log M)$, so their protocol had only two levels of recursion, whereas we have three.

6 Open Questions

One open question is to improve the number of samples. Is there an oblivious sampler that simultaneously uses a constant times optimal number of random bits and constant times optimal number of sample points? A non-oblivious sampler with these properties was constructed in [BGG93].

Another open question is whether the number of random bits in the sampler can be improved to $m + \log \gamma^{-1}$, without any constant. This would correspond to constructing an extractor that extracted all δn bits of randomness. Such an improvement would result in strong improvements in the expanders constructed in [WZ95] and in the applications there. Ta-Shma [Ta-96] can extract all the randomness by adding polylog truly random bits, which amounts to a quasi-polynomial number of samples.

Indeed, Ta-Shma's extractor works for any amount of entropy δn. This brings up another important extractor question: is there an extractor which adds $O(\log n)$ truly random bits to a δ-source with $\delta n = n^{\beta}$ and outputs $n^{\beta'}$ bits, where $0 < \beta' \leq \beta < 1$? Saks, Srinivasan, and Zhou [SSZ95] achieved the corresponding result for RP. Ta-Shma can almost construct the extractor, but is off by a factor of a logarithm iterated any constant number of times.

For leader election protocols, the natural open problem is whether similar results can be achieved for a protocol using $o(\log M)$ rounds. Perhaps an easier question is to achieve similar results with an $O(\log M)$ round protocol where each player broadcasts 1 bit per round, and not $\log M$.

Finally, it would be interesting to find more applications of extractors.

Acknowledgements

I am grateful to Noam Nisan, Amnon Ta-Shma, Aravind Srinivasan, Andrea Clementi, Russell Impagliazzo, and Ran Canetti for helpful comments and discussions. I also thank Oded Goldreich and the anonymous referees for a careful reading and many helpful comments about the write-up.

References

[AKS87] M. Ajtai, J. Komlós, and E. Szemerédi. Deterministic simulation in Logspace. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 132–140, 1987.

[AN93] N. Alon and M. Naor. Coin-flipping games immune against linear-sized coalitions. *SIAM Journal on Computing*, 22:403–417, 1993.

[AW] R. Armoni and A. Wigderson. Pseudorandomness for space-bounded computations. Unpublished manuscript.

[BGG93] M. Bellare, O. Goldreich, and S. Goldwasser. Randomness in interactive proofs. *Computational Complexity*, 3(4):319–354, 1993.

[Blu86] M. Blum. Independent unbiased coin flips from a correlated biased source: a finite Markov chain. *Combinatorica*, 6(2):97–108, 1986.

[BN] R. Boppana and B. Narayanan. Perfect-information leader election with optimal resilience. Unpublished manuscript.

[BR94] M. Bellare and J. Rompel. Randomness-efficient oblivious sampling. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 276–287, 1994.

[CEG95] R. Canetti, G. Even, and O. Goldreich. Lower bounds for sampling algorithms for estimating the average. *Information Processing Letters*, 53:17–25, 1995.

[CFG$^+$85] B. Chor, J. Friedman, O. Goldreich, J. Håstad, S. Rudich, and R. Smolensky. The bit extraction problem or t–resilient functions. In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science*, pages 396–407, 1985.

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

[CG89] B. Chor and O. Goldreich. On the power of two–point sampling. *Journal of Complexity*, 5:96–106, 1989.

[CL95] J. Cooper and N. Linial. Fast perfect-information leader-election protocol with linear immunity. *Combinatorica*, 15:319–332, 1995.

[CW89] A. Cohen and A. Wigderson. Dispersers, deterministic amplification, and weak random sources. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 14–19, 1989.

[FM97] P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous Byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, 1997.

[GG81] O. Gabber and Z. Galil. Explicit construction of linear sized superconcentrators. *Journal of Computer and System Sciences*, 22:407–420, 1981.

[Gil93] D. Gillman. A Chernoff bound for random walks on expander graphs. In *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, pages 680–691, 1993.

[GW97] 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.

[ILL89] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudorandom generation from one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 12–24, 1989.

[IZ89] 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.

[Lub86] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(4):1036–1053, 1986.

[Nis92] N. Nisan. Pseudorandom generators for space–bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[Nis96] N. Nisan. Extracting randomness: How and why – a survey. In *Proceedings of the 11th Annual IEEE Conference on Computational Complexity*, pages 44–58, 1996.

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

[ORV94] R. Ostrovsky, S. Rajagopalan, and U. Vazirani. Simple and efficient leader election in the full information model. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 234–242, 1994.

[Sak89] M. Saks. A robust noncryptographic protocol for collective coin flipping. *SIAM Journal on Discrete Mathematics*, 6:240–244, 1989.

[San87] M. Santha. On using deterministic functions in probabilistic algorithms. *Information and Computation*, 74(3):241–249, 1987.

[Sip88] M. Sipser. Expanders, randomness, or time vs. space. *Journal of Computer and System Sciences*, 36:379–383, 1988.

[SSZ95] M. Saks, A. Srinivasan, and S. Zhou. Explicit dispersers with polylog degree. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 479–488, 1995.

[SV86] M. Santha and U. V. Vazirani. Generating quasi-random sequences from semi-random sources. *Journal of Computer and System Sciences*, 33:75–87, 1986.

[SZ94] A. Srinivasan and D. Zuckerman. Computing with very weak random sources. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 264–275, 1994. Submitted for journal publication. Revised version may be obtained from **http://www.cs.utexas.edu/users/diz.**

[Ta-96] A. Ta-Shma. On extracting randomness from weak random sources. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 276–285, 1996.

[WZ95] A. Wigderson and D. Zuckerman. Expanders that beat the eigenvalue bound: Explicit construction and applications. *Combinatorica*, 1995. To appear. Revised version appears as Technical Report TR-95-21, Department of Computer Sciences, The University of Texas at Austin, June 1995. Preliminary version in *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 245–251, 1993.

[Zuc90] D. Zuckerman. General weak random sources. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 534–543, 1990.

[Zuc96] D. Zuckerman. Simulating BPP using a general weak random source. *Algorithmica*, 16:367–391, 1996.