Journal of Universal Computer Science, vol. 6, no. 1 (2000), 212-225
submitted: 15/6/00, accepted: 2/11/00, appeared: 28/1/00 © Springer Pub. Co.

Extractors for the Real World[1]

Kundi Xue
(School of Computer and Information Sciences
Georgia Southwestern State University, Americus, GA, USA
Email: kx@canes.gsw.edu.)

Marius Zimand
(Department of Computer and Information Sciences
Towson University, Towson, MD, USA
and
Department of Computer Science
University of Bucharest, Bucharest, Romania
Email: mzimand@towson.edu)

Abstract: Extractors are a special type of binary graphs that can be utilized to improve the quality of randomness sources that generate strings with small entropy. The paper explores constructions of extractors that are practical and easy to implement. Randomized and deterministic constructions are presented and compared with some previously known constructions that achieve very good asymptotical performances. One of our methods is shown to have a better behavior for reasonable values of the involved parameters.

Key Words: Random bits, source of randomness, extractors, hash functions.
Category: F.2.2, G.2.2, G.3

1 Introduction

It is commonly accepted that random bits are a valuable computational resource. Randomness has been increasingly utilized in many important areas of computer science such as simulations, algorithms, network constructions, cryptography, distributed computing, and others. Randomized algorithms may be faster, or use less memory, less processors, or less network bandwidth, or they may be simpler to implement. One major problem is the acquisition of random bits. In general the applications assume the use of perfect random bits, that is a sequence of independent bits each of them having an equal chance of being 0 or 1. In practice programmers use pseudo-random number generators and what they get has no randomness at all. There have been reports of algorithms giving quite different results under different pseudo-random generators (e.g., see [FLW92]). This has happened even for applications that do not seem to be too demanding about the quality of random bits (e.g., simulation). In some areas, such as cryptography, the quality of random bits is critical and the use of pseudo-random number generators is not considered acceptable (see [ECS94]). A better alternative is to

[1] C. S. Calude and G. Ştefănescu (eds.). *Automata, Logic, and Computability. Special issue dedicated to Professor Sergiu Rudeanu Festschrift.*

use a natural source of randomness (Geiger counters and Zener diodes have been the most commonly cited in this role). Such sources contain real randomness, but not of sufficient quality for sensitive applications.

An extractor is a combinatorial object that is used to improve the quality of a random source. For the sake of illustration, let us suppose that we have a natural source of randomness (e.g., a Geiger counter measuring radioactivity emissions) that generates a string of n bits. We would like each string to have a probability of 2^{-n} of being produced (this means perfect randomness). However, randomness sources typically have some biases and they allocate more probability mass to some strings (and of course less to some others). The exact qualitative formulation of this phenomenon is captured by the concept of *maximum mass* of a probability distribution (similar in many aspects to the entropy of the distribution) and which is defined as follows. Let D be the probability distribution associated to a randomness source for binary strings of length n. The maximum mass of D is defined by max-mass$(D) = \max D(x)$, where the maximum is taken over all the strings of length n. Note that the bigger max-mass(D) is, the more defective the source is. We can now define the main object of interest in this paper.

Definition 1 *An (n, k, d, m, ϵ) extractor is a bipartite regular multigraph $G = (V_{left}, V_{right}, E)$, where V_{left} has 2^n nodes identified with the strings of length n, V_{right} has 2^m nodes identified with the strings of length m, and with the degree of each node in V_{left} equal to 2^d; it has the property that if x is chosen randomly in V_{left} according to a distribution D with max-mass(D) at most 2^{-k}, and if y is chosen uniformly at random among the 2^d edges outgoing from x, the distribution of the string $(E(x,y), y)$ is ϵ-close to the uniform distribution.*[2]

In other words, we start with an n-bit string x provided by a defective source, we process it using a perfectly random string y of length d to obtain a string $E(x, y)$ of length m, and then we append to it y (so that the good random strings are not lost). What we obtain is very close (within distance ϵ) to a perfectly random string.

It is not surprising that extractors have attracted much attention. The main objective has been the construction of extractors with better parameters. In general, considering n and k as fixed, we want to make d and ϵ small and m large. The best results have been established by Zuckerman [Zuc97], Ta-Shma [TS96], Trevisan [Tre99], and Raz, Reingold, and Vadhan[RRV99]. In the case in which $k = \Omega(n)$, Zuckerman has constructed an extractor with $d = O(\log(n) + log(\epsilon^{-1}))$ and $m = \Omega(n)$. Ta-Shma's extractor works for any k (of course, $k \leq n$) and has $d = poly(\log(n) + \log(\epsilon^{-1}))$ and $m = k$. Trevisan's extractor works with

[2] Two distributions D_1 and D_2 on the same sample space S are ϵ-close if $\sum_{x \in S} |D_1(x) - D_2(x)| \leq \epsilon$.

$k = n^{\Omega(1)}$ and has $m = k^{\Omega(1)}$ and $d = O(\log n)$ for constant ϵ. The extractor of Raz, Reingold, and Vadhan works for any k and achieves $m = k^{1-\alpha}$ and $d = O(\log^2(n) \cdot \log(1/\epsilon)/\log k)$. These results are undeniably impressive and they have lead to breakthrough results in some applications of extractors. On the other hand, these extractors are the result of a sophisticated amalgamation of some more basic extractors and of some other combinatorial objects and, as a consequence, the hidden constants appear to be very large. Also, while the computation of $E(x,y)$ is doable in polynomial time in all the constructions cited above, they are by no means practical.

We investigate here constructions of extractors that are simple, efficient, easy to program, and whose parameters are good in a realistic setting, this meaning for reasonable values of n and m such as n= 1024, 512, or 256 and m = 512, 256, or 128, which in our opinion should be interesting in practice. We first observe that random bipartite graphs of a certain type are with high probability good extractors. The drawback is that checking if a graph is an extractor is NP complete and thus we cannot obtain any guarantee about the quality of such an extractor. We next show that a random bipartite graph of another type meets with high probability a condition that can be checked in polynomial time and that is sufficient to make the graph a good extractor. However, in practice, the condition can be checked only for small values of n and m. Next we consider deterministic constructions. A main ingredient in many of these constructions is the utilization of hashing functions with small collision error. Many hashing functions with 0 collision error are known, but, in this setting, the challenge is to find hashing functions that can be specified with a small number of bits, because the length of the hashing function is exactly the parameter d in the definition of an extractor. With the issue of practicality in mind, we design a new family of hashing functions that are specified with $d = 2m(\log(n/(2m)) + 1/2)$ bits, have $\alpha \approx 2^{-m}$ collision error, and have an efficient implementation. They yield immediately an $(n, k, d, m, \sqrt{\alpha + 2^{-(k-m)}})$ extractor. We compare our construction to one of Srinivasan and Zuckerman [SZ94] which has been used as a basic element in Zuckerman's and Ta-Shma's constructions cited above. While asymptotically somewhat weaker, for reasonable values of n and m, our hashing functions need a number of random bits that is close and in many cases even smaller than the requirements in [SZ94]. Our method is easy to implement and from this point of view is vastly superior to the other method.

2 Randomized constructions

One possibility is to do a randomized construction. This is a consequence of the following result. (Similar results for a related type a graphs - dispersers - have been established by Sipser [Sip86].)

Proposition 1. *Let us consider the family $\mathcal{G}$ of graphs G of the form $G = (V_{left}, V_{right}, E)$, with $|V_{left}| = N = 2^n, |V_{right}| = M = 2^m$ and degree $D = 2^d = n \cdot \epsilon^{-2}$, where $m = \alpha n$ for some $\alpha < 1$. Then, with high probability, a randomly chosen graph from $\mathcal{G}$ is an (n, m, d, m, ϵ)-extractor.*

Proof. Since $E(x, y)$ and y are chosen independently, it is enough to show that the distribution of $E(x, y)$ is ϵ-close to the uniform distribution when x is chosen from V_{left} according to a distribution D_ℓ with max-mass$(D_\ell) \leq 2^{-m}$ and y is chosen uniformly at random from $\{0, 1\}^d$. It can be shown that we can consider a flat distribution in the role of D_ℓ, i.e., we can consider that D_ℓ assigns 2^{-m} probability mass to 2^m nodes from V_{left} and zero probability mass to the rest of the nodes. Thus, it remains to show that, with high probability, for any set A_ℓ of M nodes from V_{left}, the distribution of $E(x, y)$ is ϵ-close to the uniform distribution, when x and y are chosen randomly from A_ℓ and respectively $\{0, 1\}^d$. Let us denote the above event by GOOD. $E(x, y)$ is ϵ-close to the uniform distribution if for all subsets A_r of V_{right},

$$\left| Prob(E(x, y) \in A_r) - \frac{|A_r|}{|V_{right}|} \right| < \epsilon.$$

If the above relation holds for a set A_r, then it also holds for its complement. So, it is enough to restrict our attention to sets A_r with $|A_r| \leq M/2$.

Let us choose a subset $A_\ell \subseteq V_{left}$ with $|A_\ell| = M$ and a subset $A_r \subseteq V_{right}$ with $|A_r| = P \leq M/2$. For each $x \in A_\ell$ and $y \in \{0, 1\}^d$, let us call $E(x, y)$ to be a *throw*. Then $Prob(E(x, y) \in A)$ is equal to (number of throws that hit A_r)/ (numbers of throws). The probability that a throw hits A_r is P/M and there are MD throws. Let X_i, $i = 1, \ldots, MD$, be a random variable, which is 1 if the i-th throw hits A_r, and 0 otherwise. Let

$$S = \sum_{i=1}^{MD} X_i.$$

Since the variables X_i are independent, the Chernoff bound yields

$$\begin{aligned} Prob\left(\left|\frac{S}{MD} - \frac{P}{M}\right| > \epsilon\right) &< 2e^{-2\epsilon^2 MD} \\ &= 2e^{-2nM}. \end{aligned}$$

The second line is due to the fact that $D = n \cdot \epsilon^{-2}$. A_ℓ can be chosen in $\binom{N}{M}$ ways, and A_r can be chosen in $\binom{M}{P}$ ways. Thus the probability that there is some A_ℓ and some A_r as above such that the probability of a throw from A_ℓ hitting A_r is not within ϵ distance from the probability of A_r is bounded from above by

$$\binom{N}{M}\binom{M}{P} 2e^{-NM} \leq \left(\frac{eN}{M}\right)^M \left(\frac{eM}{P}\right)^P \cdot e^{-2NM}$$

$$
\begin{aligned}
&< e^{M+P} \cdot 2^{M \log N} \cdot 2^{-M \log M} \cdot 2^{P \log M} \cdot e^{-2nM} \\
&< e^{-(2nM + M \log M - M - P - M \log N - P \log M)} \\
&<< \frac{1}{M^2}.
\end{aligned}
$$

Since P is chosen from $\{1, \ldots, M/2\}$, the complement of GOOD has probability much less than $(M/2) \cdot M^{-2} << 1$. ∎

Thus it is enough to link randomly each node from V_{left} to D nodes in V_{right} and the graph that is obtained is with high probability a pretty good extractor. Unfortunately, if we want to be guaranteed that the random outcome is indeed an extractor, then we have a problem: checking whether a graph is an extractor is NP-complete and the problem seems to be intractable even for moderate values of n and m.

Proposition 2. *Checking whether a graph is an extractor is* NP *complete.*

Proof. We reduce the *Regular Matcher* problem to "Checking whether a bipartite graph is an $(n, k, d, m, 1/p)$ extractor." The *Regular Matcher* problem has been shown to be NP complete in [BKV+81] and is defined as follows:

Input: A bipartite graph $G = (V_1, V_2, E)$ with $|V_1| = |V_2| = 2M$.

Question: Is it the case that for any $S \subset V_1$, $|S| = M$, there exists $T \subset V_2$ such that $|S| = |T|$ and there is a matching between S and T?

We will use the fact that a bipartite graph $G = (V_1, V_2, E)$ with $|V_1| = |V_2| = 2M$ is a regular matcher if and only if for each $S \subset V_1$ with $|S| = M$, $|\Gamma(S)| \geq |S|$, where $\Gamma(X)$, denotes the set of nodes adjacent in G to the nodes in the set X.

The reduction works as follows. Let $G = (V_1, V_2, E)$ be a bipartite graph with $|V_1| = |V_2| = 2M$. We can assume that G is regular, since otherwise we can increase the number of outgoing edges from a node $x \in V_1$ by adding some multiple edges outgoing from x (i.e., if (x, y) is an existing edge, we add some copies of it.) Let t be the degree of each node in V_1. We build a bipartite graph $G' = (V_1', V_2', E')$ as follows: $V_1' = V_1$; $V_2' = V_2 \cup W$, where W is a set of new nodes and $|V_2'| = tM + 1$; and $E' = E$. We show that G' is a $(2M, \log M, t, tM + 1, 2(1 - M/(tM + 1)))$ extractor if and only if G is a regular matcher.

Let G be a regular matcher and suppose that G' is not an extractor of the required type. Then there is a distribution D on V_1 with max-mass$(D) \leq 1/M$ such that $dist(E(D, U_t), U_{V_2'})$, the distance between the distribution $E(D, U_t)$ and the uniform distribution $U_{V_2'}$ on V_2', is greater than 2ϵ, where $\epsilon = (1 - M/(tM + 1))$. ($E(D, U_t)$ is the distribution induced on V_2' by choosing randomly according to D a node in V_1 and then following the i-th outgoing edge with i chosen uniformly at random from $T = \{1, \ldots, t\}$). It can be shown that D can be taken to be a flat distribution, i.e., there is a set $X \subset V_1$ with $|X| = M$ such that D assigns $1/M$ probability mass to the elements in M and 0 probability mass to the nodes in $V_1 - M$. Let us denote $E(D, U_t)$ by D_1 and U_{V_2} by D_2. It is easy to see that

$$dist(D_1, D_2) = 2(Prob_{D_2}(A) - Prob_{D_1}(A)),$$

where $A = \{x \, : \, D_2(x) > D_1(x)\}$ (this holds for any distributions D_1 and D_2). The distribution D_1 either places zero probability mass on an element in V_2' or else it places probability mass at least

$$\frac{1}{|X||T|} = \frac{1}{tM} > \frac{1}{|V_2'|}.$$

It follows that there is a set $K \subseteq V_2'$ such that

$$Prob_{D_1}(K) = 0$$

and

$$Prob_{D_2}(K) > \epsilon.$$

Consequently, $|K| > \epsilon|V_2|$ and $\Gamma(X) \cap K = \emptyset$. Then,

$$\Gamma(X) < (1-\epsilon)|V_2'| = \frac{M}{tM+1}(tM+1) = M = |X|,$$

which contradicts the fact that G is a regular matcher.

Conversely, suppose G' is an extractor with the required parameters. If G is not a regular matcher, then there is $X \subset V_1$, $|X| = M$, and $|\Gamma(X)| < |X| = M$. Let $H = V_2' - \Gamma(X)$. Then

$$|H| > tM + 1 - M = \epsilon|V_2'|.$$

Thus,

$$Prob_{U_{V_2'}}(H) > \epsilon$$

and

$$Prob_{E(U_X, U_t)}(H) = 0,$$

where U_X is the uniform distribution on X. This contradicts the fact that G' is an extractor with the required parameters. ∎

Let us therefore explore other possibilities. Hash functions are a key ingredient in many of the known constructions of extractors. The basic technical tool in these constructions is the leftover hash lemma (see [ILL89], [Nis96]). We shall use a variant of it. First we need some definitions.

Definition 3. Let H be a family of functions $h : \{0,1\}^n \to \{0,1\}^m$. We say that H has ϵ collision error if for all $x_1 \neq x_2$, $Prob_{h \in H}(h(x_1) = h(x_2)) \leq (1+\epsilon)2^{-m}$.

Definition 4. The collision probability of a distribution D on $\{0,1\}^n$ is $Prob_D(X = Y) = \sum_{x \in \{0,1\}^n} D^2(x)$, where X and Y are independent random variables having distribution D.

The following facts are well known. We include their proofs for completeness.

Lemma 5. *Let D be a distribution on $\{0,1\}^n$ with collision probability at most $2^{-n}(1+\epsilon^2)$. Then D is ϵ-close to the uniform distribution on $\{0,1\}^n$.*

Proof. .

$$\sum_{x \in \Sigma^n} |D(x) - 2^{-n}| \leq \sqrt{2^n} \sqrt{\sum_{x \in \Sigma^n} (D(x) - 2^{-n})^2} \text{ (Cauchy-Schwartz inequality)}$$
$$= \sqrt{2^n} \sqrt{\sum_{x \in \Sigma^n} D^2(x) + \sum_{x \in \Sigma^n} 2^{-2n} - 2 \cdot 2^{-n} \sum x \in \Sigma^n D(x)}$$
$$\leq \sqrt{2^n} \sqrt{2^{-n}(1+\epsilon^2) - 2^{-n}} = \epsilon.$$

∎

Lemma 6. *Let D be a distribution on $\{0,1\}^n$ with max-mass$(D) = 2^{-k}$. Then the collision probability of D is upper bounded by 2^{-k}.*

Proof. The collision probability of D is $\sum_{x \in \Sigma^n} D^2(x)$. Since $\sum_{x \in \Sigma^n} D(x) = 1$ and $D(x) \leq p/2^n$, this expression is maximized for distributions D allocating $p/2^n$ probability mass to $2^n/p$ elements in Σ^n and 0 to the rest of the elements. In this case, the collision probability is $2^n/p$. ∎

The following is a variant of the left-over hash lemma.

Lemma 7. *Let H be a family of hash functions $h : \{0,1\}^n \to \{0,1\}^m$ with ϵ collision error. Then the distribution of $(h(x), h)$, when x is chosen from $\{0,1\}^n$ according to a distribution with max-mass $\leq 2^{-k}$ and h is chosen uniformly at random in H, is $\sqrt{\epsilon + 2^{-(k-m)}}$-close to the uniform distribution.*

Proof. We evaluate the collision probability of $(h(x), h)$.

$$\begin{aligned}
&Prob_{x,h,x',h'}((h(x),h) = (h'(x'),h')) \\
&= Prob_{h,h'}(h = h') Prob_{x,x',h}(h(x) = h(x')) \\
&= \frac{1}{|H|}(Prob_{x,x'}(x = x') + Prob_{x,x',h}(h(x) = h(x') \mid x \neq x')). \qquad (1)
\end{aligned}$$

The first term is bounded by 2^{-k} (we have used the hypothesis on the maximum mass of the distribution of x and Lemma 6). We evaluate the second term.

$$
\begin{aligned}
&Prob_{x,x',h}(h(x) = h(x') \mid x \neq x') \\
&= \sum_{u \neq u'} Prob_h(h(u) = h(u')) Prob_{x,x'}(x = u \text{ and } x' = u' \mid x \neq x') \\
&\leq (1+\epsilon)2^{-m} \sum_{u \neq u'} Prob_{x,x'}(x = u \text{ and } x' = u' \mid x \neq x') = (1+\epsilon)2^{-m}.
\end{aligned}
$$

We have used the fact that H has ϵ collision error. It follows that equation (1) is bounded from above by

$$\frac{1}{|H|}(2^{-k} + (1+\epsilon)2^{-m}) = \frac{1}{|H| \cdot 2^m}(1 + 2^{-(k-m)} + \epsilon).$$

Taking into account Lemma 5, the conclusion follows. ∎

It turns out that a random regular bipartite graph yields with high probability a family of hash functions with small collision error. More precisely, the following theorem holds.

Theorem 8. *Let $G = (V_1, V_2, E)$ be a random regular bipartite graph with the left-hand side $V_1 = \{0,1\}^n$, the right-hand side $V_2 = \{0,1\}^m$, and degree $D = 2^d = 9n \cdot 2^m \cdot 1/(\epsilon^2)$. For each $h \in \{0,1\}^d$ we define a hash function $h : \{0,1\}^n \to \{0,1\}^m$ by $h(x) = E(x,h)$, where $E(x,h)$ is the node from V_2 that is obtained from x following the edge labeled with h. Let H be the family of functions h. Then with probability of G at least $1 - 2^{-n}$, H has ϵ collison error.*

Proof. For fixed $a, a' \in \{0,1\}^n$, with $a \neq a'$, and for each $h \in \{0,1\}^d$, let X_h be 1, if $E(a,h) = E(a',h)$, and 0, otherwise. Clearly, $Prob(X_h = 1) = 2^{-m}$ and the random variables X_h are independent. By Chernoff bounds, $Prob_G((\sum X_h)/D \geq 2^{-m}(1+\epsilon)) \leq e^{-(1/3)\cdot\epsilon^2 \cdot D \cdot 2^{-m}} < 2^{-3n-1}$. Thus, the fraction of edges h such that $E(a,h) = E(a',h)$ is $\geq 2^m(1+\epsilon)$, only with probability of G less than 2^{-3n-1}. The probability that there is a pair a, a' as above is less than $2^{-(n+1)}$. The conclusion follows. ∎

Thus, if we build $D = 9n \cdot 2^m \cdot 1/(\epsilon^2)$ random functions $h : \{0,1\}^n \to \{0,1\}^m$ we obtain a family H that, with high probability, can be used to define a good extractor. Calculating the collision error of H can be done in time polynomial in $N = 2^n$ (the number of nodes in V_1), $M = 2^m$ (the number of nodes in V_2), and ϵ^{-1}. Thus, it seems that one has to repeatedly build families H of random functions and check if the family has the desired collision error. One may hope to fall upon on a good family H even when trying a smaller D than the one prescribed by Theorem 8. Unfortunately, the time for calculating the collision error is quadratic in 2^n and, consequently, this operation can be performed only for quite small values of n. Also, in our experiments, smaller values of D did not produce a reasonable ϵ (i.e., in the order of 2^{-m}). In our experiments we have used the collection of random bits provided by Marsaglia [Mar96]. Marsaglia has produced high-quality random bits using a combination of random sources (three sources of electronic white noise) and pseudo-random number generators. The bits have passed the state-of-the-art tests of randomness. These bits are available in form of a CD-ROM (or they can be downloaded from the web [***96]) containing sixty 10-megabyte files. We have used these bits to obtain the families H of random functions and then we have calculated the collision error. Some of the (disappointing) results are given in Table 1 (time represents the running time for calculating ϵ on a Pentium 200 MHz machine).

n	m	D	ϵ	time (in sec.)
7	4	8000	0.164	
7	4	10000	0.136	
8	5	8192	0.273	58299
8	5	12000	0.216	84220

Table 1: Calculation of the collision error for a random family of hash functions

3 Deterministic constructions

Let us turn to deterministic constructions. As we have mentioned in the Introduction, the best constructions known at this moment are not practical. A basic element in many of these constructions is a simpler extractor designed by Srinivasan and Zuckerman [SZ94], which is based on a family of hash functions. They have constructed such a family H of functions $h : \{0,1\}^n \to \{0,1\}^m$, where each h is specified by $d = 2(\log(n+\log(m))+\log(m)+2m+2\log(\epsilon^{-1}))$ bits, that has ϵ-collision error. By Lemma 7, this immediately yields an $(n, k, m, d, \sqrt{\epsilon + 2^{-(k-m)}})$ extractor. For the sake of comparing with the method that we have developed and which will be described later, we describe the construction of Srinivasan and Zuckerman. Their method relies on a set of "almost k-wise independent random variables" constructed by Alon et al. [AGHR92].

Input: n, m and ϵ.

Step 1. Choose two strings x and y of length r, with

$$r \geq \log(n + \log m) + \log(m) + 2m + 2\log(1/\epsilon).$$

Also, r should be a power of two. The two strings x and y define a hash function h from the family H.

Step 2. Construct the string

$$R = (x \cdot y) \circ (x^2 \cdot y) \circ \ldots \circ (x^{m(n+\log(m))} \cdot y),$$

where the exponentiations are done in the field $GF(r)$, $\cdot$ denotes inner product modulo 2, and $\circ$ denotes concatenation.

Step 3. Let L be the parity check matrix of BCH codes, of size $m(n+\log m)\times \mathbf{2}m$. In a more detailed manner, L is defined as follows. Let u be the smallest integer so that

$$2^u \geq 2^n m + 1.$$

Let $x_1, \ldots, x_{2^u-1}$ be the nonzero elements of the field $GF(2^u)$. Using the standard representation of $GF(2^u)$, we view each x_i as a column vector with u binary entries. Let t be the smallest integer such that $1 + ut \geq m(n + \log m)$. Then

$$L = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ x_1 & x_2 & \ldots & x_{2^u-1} \\ x_1^3 & x_2^3 & \ldots & x_{2^u-1}^3 \\ \vdots & & & \\ x_1^{2t+1} & x_2^{2t+1} & \ldots & x_{2^u-1}^{2t+1} \end{pmatrix}$$

The hashing function $h : \{0,1\}^n \to \{0,1\}^m$ is defined by

$$h(\underbrace{00\ldots0}_{n}) \circ h(\underbrace{00\ldots1}_{n}) \circ h(\underbrace{11\ldots1}_{n}) = R \cdot L.$$

(In fact, to be accurate, in the right hand side we take only the first $2^n m$ entries of $R \cdot L$ which will represent the concatenation of all the hashed via h images of the strings in $\{0,1\}^n$.)

The family H of these hashing functions has ϵ collision error.

Since the matrix L has at least $2^n m$ columns, it cannot be calculated explicitly except for very small values of n and m. However, note that in order to calculate $h(a)$, for some $a \in \{0,1\}^n$, it is not necessary to have the whole L. We split the columns of L into consecutive blocks of m columns and we denote by $slice(a)$ the a-th block. Then $h(a) = R \cdot slice(a)$. Thus, it is enough to calculate the a-th block. However, this still requires the calculation of an $m(n+\log m) \times m$ matrix, which is a very time consuming operation that has to be performed each time one hashes a point a.

We consider a new family H of hash functions $h : \{0,1\}^n \to \{0,1\}^m$ having

$$\alpha = (\log n - \log m)2^{-m} - 2^{-3m}$$

collision error. A function h in the family is specified by a number of bits equal to

$$\delta = 2m \cdot (\log(n/(2m)) + 1/2).$$

Again, by Lemma 7, this immediately yields an $(n, k, m, \delta, \sqrt{\alpha + 2^{-(k-m)}})$ extractor. The description fits into a few lines:

Input: n, m.

Step 1. Let $s = \log(n/(2m))$. The hash function h is given by $s + 1$ strings, $y_1, \dots, y_s$ and z, where each y_i has length $2m$ and z has length m. The string a of length n that will be hashed is viewed as a polynomial p_a of s variables, of degree one in each variable, over the field $GF(2^{2m})$. In this step, we calculate $p_a(y_1, \dots, y_s)$.

Step 2. The output of the first step gives the two coefficients of a linear function $l(x) = cx + d$ over $GF(2^m)$. We output $l(z)$ which represents $h(a)$.

Let us prove that this construction achieves the claimed performance.

Theorem 9. *The family H of hash functions $h : \{0,1\}^n \to \{0,1\}^m$ defined above has $\alpha = (\log n - \log m)2^{-m} - 2^{-3m}$ collision error.*

Proof. Let $k = 2m$ and $\overline{y} = (y_1, \dots, y_s)$. In the first step, we take the point $a \in \{0,1\}^n$ that will be hashed and we split it into 2^s substrings $a_1, \dots, a^{2^s}$, each of size k. These substrings define the polynomial

$$p_a(x_1, \dots, x_s) = a_0 + a_1 x_1 + \dots a_{2^s} x_1 \dots x_s$$

of degree one in each variable. For $a_i \neq a_j$, let

$$Col(i,j) = \{\overline{y} \in \{0,1\}^{sk} \ : \ p_{a_i}(\overline{y}) = p_{a_j}(\overline{y})\}.$$

By induction on s, it can be shown that

$$|Col(i,j)| \leq sK^{s-1} - K^{s-2}, \text{ for } s \geq 2,$$

where $K = 2^k$. Therefore,

$$Prob(\overline{y} \in Col(i,j)) \leq \frac{sK^{s-1} - K^{s-2}}{K^s} = \frac{s}{2^{2m}} - \frac{1}{2^{4m}}. \tag{2}$$

In the second step, $p_a(\overline{y})$ is divided into 2 blocks, c and d, each of length m, and we calculate $cz + d$ in $GF(2^m)$. Let us denote c by $p_a^1(\overline{y})$ and d by $p_a^2(\overline{y})$. The collision error is given by

$$\max_{a_1 \neq a_2} Prob_{h=(y_1,\dots,y_s,z)}(h(a_1) = h(a_2)).$$

For any pair $a_1 \neq a_2$,

$$\begin{aligned}&Prob_{h=(y_1,\ldots,y_s,z)}(h(a_1) = h(a_2)) = \\ &Prob_{h=(y_1,\ldots,y_s,z)}(p_{a_1}(\overline{y}) = p_{a_2}(\overline{y})) + \\ &Prob_{h=(y_1,\ldots,y_s,z)}(p^1_{a_1}(\overline{y})z + p^2_{a_1}(\overline{y}) = p^1_{a_2}(\overline{y})z + p^2_{a_2}(\overline{y}) \mid p_{a_1}(\overline{y}) \neq p_{a_2}(\overline{y})).\end{aligned}$$

From (2), the first term is bounded by $s \cdot 2^{-2m} - 2^{-4m}$. For the second term, it is enough to observe that a non-null polynomial degree of degree at most one has at most one root. It follows that the second term is bounded by 2^{-m}. Thus, for each $a_1 \neq a_2$,

$$\begin{aligned}Prob_{h=(y_1,\ldots,y_s,z)}(h(a_1) = h(a_2)) &\leq s2^{-2m} - 2^{-4m} + 2^{-m} \\ &= 2^{-m}(1 + s2^{-m} - 2^{-3m}),\end{aligned}$$

and thus the collision error is bounded by $s2^{-m} - 2^{-3m}$. ∎

It is noteworthy the following twist of roles in comparison with traditional hashing methods: the point that is hashed provides a polynomial function (p_a) and the hashing function gives the elements that are mapped via this function. Even if asymptotically somewhat weaker, for realistic values of n and m, our construction needs a number of random bits that is close and in many cases smaller than the requirements of the method in [SZ94] (see Table 2; the collision error has been chosen to be the same for the two methods, namely $(\log n - \log m)2^{-m} - 2^{-3m}$, which explains why the number of bits is not monotonically increasing with m for the [SZ94] method).

n	m	no. of random bits	
		this paper	[SZ94] paper
512	64	320	538
512	128	384	1056
1024	128	640	1054
1024	256	768	2084
2048	64	576	538
2048	128	896	1053
2048	256	1280	2082
2048	512	1536	4136

Table 2: Comparison of the number of random bits. The collision error is $(\log n - \log m)2^{-m} - 2^{-3m}$.

The main goal of the design has been to facilitate as much as possible the implementation while maintaining competitive values of the parameters. For example, the polynomials p_a could have been taken of degree s in each variable. This would have reduced the number of random bits but would have made the evaluation of $p_a(\overline{y})$ more difficult. As a result, the running time outperforms by far the method of [SZ94] (see Table 3; ; for the algorithm in [SZ94] we have taken the collision error to be 0.01, while for our method the collision error is $(\log n - \log m)2^{-m} - 2^{-3m}$, a value that is much smaller than 0.01).

n	m	running time for 10 iterations	
		this paper	[SZ94] paper
256	64	48 ms	341750 ms
512	64	54 ms	895600 ms
1024	128	220 ms	8750230 ms
1024	256	390 ms	44376580 ms

Table 3: Comparison of running times. The collision error is $(\log n - \log m)2^{-m} - 2^{-3m}$.

References

[***96] ***. Marsaglia's random bits. `ftp://ftp.cs.hku.hk/pub/random`, 1996.

[AGHR92] N. Alon, O. Goldreich, J. Håstad, and R.Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures and Algorithms*, 3(3):289–304, 1992.

[BKV+81] M. Blum, R.M Karp, O. Vornberger, C.H. Papadimitriou, and M. Yannakakis. The complexity of checking whether a graph is a superconcentrator. *Information Processing Letters*, 13(4,5):164–167, 1981.

[ECS94] D. Eastlake, S. Crocker, and J. Schiller. RFC 1750 - Randomness requierements for security. Internet Request for Comments 1750, December 1994.

[FLW92] A.M Ferrenberg, D.P. Landau, and Y.J. Wong. Monte Carlo simulations: hidden errors from "good" random number generators. *Physical Review Letters*, 69(23):3382–3384, 1992.

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

[Mar96] G. Marsaglia. Diehard. `http://stat.fsu.edu/~geo/diehard.html`, 1996.

[Nis96] N. Nisan. Extracting randomness: how and why. A survey. In *Proceedings of the 11th Structure in Complexity Theory Conference*, pages 44–58, 1996.

[RRV99] R. Raz, O. Reingold, and S. Vadhan. Extracting all the randomness and reducing the error in trevisan's extractor. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 149–158. ACM Press, May 1999.

[Sip86] M. Sipser. Expanders, randomness, or time versus space. In *Proceedings of the 1st Structure in Complexity Theory Conference*, pages 325–329. Springer Verlag *Lecture Notes in Computer Science #223*, June 1986.

[SZ94] A. Srinivasan and D. Zuckerman. Computing with very weak random sources. In *Proceedings of the 34th IEEE Symposium on Foundations of Computer Science*, pages 264–275, 1994.

[Tre99] L. Trevisan. Constructions of near-optimal extractors using pseudo-random generators. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 141–148. ACM Press, May 1999.

[TS96] A. Ta-Shma. On extracting randomness from weak random sources. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 276–285, 1996.

[Zuc97] D. Zuckerman. Randomness-optimal oblivious sampling. *Random Structures and Algorithms*, 11:345–367, 1997.