How to Recycle Random Bits

Russell Impagliazzo * David Zuckerman †

Abstract

We show that modified versions of the linear congruential generator and the shift register generator are provably good for amplifying the correctness of a probabilistic algorithm. More precisely, if r random bits are needed for a BPP algorithm to be correct with probability at least 2/3, then $O(r + k^2)$ bits are needed to improve this probability to $1 - 2^{-k}$. We also present a different pseudo-random generator that is optimal, up to a constant factor, in this regard: it uses only $O(r + k)$ bits to improve the probability to $1 - 2^{-k}$. This generator is based on random walks on expanders. Our results do not depend on any unproven assumptions.

Next we show that our modified versions of the shift register and linear congruential generators can be used to sample from distributions using, in the limit, the information-theoretic lower bound on random bits.

1. Introduction

Randomness plays a vital role in almost all areas of computer science, both in theory and in practice. Randomized algorithms are often faster or simpler than the deterministic algorithms for the same problem. Besides speeding up computation, there are also many circumstances where one needs to sample from some probability distribution. For example, a cryptographic protocol might require a random prime or list of primes; a scientific simulation modelling a probabilistic or chaotic process (e.g. the weather) will require randomness.

Since random bits are expensive, programmers attempt to minimize the amount of random bits actually used through the utilization of pseudo-random generators (prg's). A deterministic process is performed on a short random "seed" to produce a much longer output. It is hoped that this longer output will serve the same purpose as a truly random string of the same size. The most commonly used prg is some sort of linear-congruential generator.

Recent theoretical work has provided some good ways of conserving random bits. Blum and Micali [BM] introduced the notion of a cryptographically secure prg; Yao [Y] showed that such a generator produces output strings which are computationally indistinguishable (in polynomial time) from truly random strings.

Nevertheless, there are problems with using cryptographically secure prg's. First, such generators must be based on the existence of one-way functions, an unproven assumption. Second, even assuming that the function in question is one-way, the performance of such prg's is only guaranteed asymptotically. In a particular example, one cannot compute exact bounds on the size of the seed needed. Third, most such generators are too inefficient to be used in practice. Linear congruential generators, although known not to be cryptographically secure (see e.g. [FKL],[P]), continue to be used in practice.

Because of these disadvantages, work has been done to construct good generators for more specific tasks. For example, Santha [Sa] and Sipser [Si] introduced the notion of a "quasi-perfect pseudo-random generator." A quasi-perfect prg can be used to decrease the probability of error of a BPP or RP algorithm from a constant $< 1/2$ to an exponentially small amount, using only a constant factor more random bits. Until now, no prg was proven to be quasi-perfect, although Vazirani [V] conjectured that a specific prg, related to the shift register prg presented here, is quasi-perfect.

Another recent direction of research has been to theoretically explain the practical success of simple prg's like the linear congruential generator. Bach [B] and Karloff and Raghavan [KR] have shown that linear congruential generators or variants thereof have provably good performances when used in specific algorithms, such as taking square roots modulo a prime and Quicksort; the latter paper also gives a task for which the usual linear congruential generators are inadequate.

We combine these two directions of research by proving that two very simple types of prgs, one of which is just a minor modification of the linear congruential generator, are quasi-perfect. Thus, there is a theoretical justification for the widely held belief that very simple, computationally predictable generators (such as linear congruential generators) still work well in practice.

Sipser has shown that the existence of certain types of constructive expanders implies that of quasi-perfect prg's; our results could be rephrased as explicit constructions of Sipser expanders (with somewhat weaker parameters than those used in [Si]).

The first construction involves taking a random walk on a type of expander graph known to be constructible [GG]. The proof, which was independently discovered by Cohen and Wigderson [CWi], uses techniques similar to those in [AKS]. These quasi-perfect prg's can be implemented simply and efficiently using explicit constructions of expanders from [GG] or [LPS]. They yield almost optimal trade-offs between the amount of randomness used and the probability of error; given an algorithm which uses r random bits and has a constant $< 1/2$ error probability, the probability of error can be reduced to 2^{-k} using $O(r + k)$ random bits.

*Supported by NSF Grant CCR88-13632. Current address: Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M6G1Y1

†Supported by an NSF Graduate Fellowship. Current address: Division of Computer Science, University of California, Berkeley, CA 94720

The second class of quasi-random prg's can be based on any family of universal hash functions ([CW]). In particular, we prove that modified versions of both the linear congruential and shift register generators (from [V]) are quasi-perfect. The modification is that only the $r - O(k)$ least significant bits of the generator's output on an r bit seed are used; the remaining bits are replaced by truly random bits. This method uses $O(r + k^2)$ random bits to reduce the error probability to 2^{-k}.

The proof is based on a lemma from [ILL], and is just one application of a general technique which can be used to reduce the random bits used in computation. The intuition behind this result is that, when we run a probabilistic algorithm for a language, the only relevant information we need to see about the random string used is the single bit, "Does the algorithm accept using this string?" Thus, the entropy of the conditional distribution (given this one-bit output) is still large. We give a way of "recycling" this entropy for future use. This technique can be used in many situations where an algorithm's output has a smaller entropy than the number of random bits it uses.

For example, we can use it to sample many times from any polynomial-time samplable distribution using the information theoretic lower bound on random bits. More concretely, suppose we wish to generate at random n bit primes (for use in a cryptographic protocol, say). A naive algorithm might work as follows. Pick a random n bit number. Use a probabilistic algorithm (e.g. Rabin's[R]) to test whether the number you picked is prime. If so, output it. If not, start over. This algorithm uses an expected $O(n^2 + nt(n))$ random bits, where $t(n)$ bits are used in the primality test, since a $O(1/n)$ fraction of n bit numbers are prime. For the same reason, information-theoretically, at least $n - \log n + c$ bits are required to generate a random prime. We show that, if a list of many primes is desired (here, "many" means a fixed polynomial in n), an amortized cost of the exact information theoretical lower bound, i.e. $n - \log n + c$ random bits per prime, is obtainable efficiently. (The output will deviate from a truly random sequence of primes by an exponentially small amount.) This can be construed as an effective version of Shannon's information theoretic result that $Ent(D)$ is equal (within a bit) to the minimum expected number of random bits for a method of sampling from D [Sh].

We generalize this result still further. In order to get the amortized savings in random bits, it is not necessary to always sample from the same distribution D. An arbitrary sequence of distributions can be sampled from; the next distribution in the sequence can depend in an arbitrary way on the outputs of the sampler on the preceding distributions. This method could be used by an operating system in order to run many probabilistic algorithms using as few random bits as possible, or it could be used to reduce the total random bits used by a probabilistic algorithm which has a subroutine or subroutines that use more random bits than they have outputs. The techniques presented here can be used to reduce the amount of randomness needed in specific algorithms for many types of problems (decision problems, approximate counting, generating elements from a distribution) even if the problems aren't explicitly in one of the forms mentioned earlier. As an example, in the final version of this paper, we will show how to generate a random n-bit prime within a statistical error 2^{-k} using $O(kn)$ bits.

2. PRG Based on Expanders

BPP is the set of languages $L \subseteq \{0,1\}^*$ such that there is a deterministic polynomial time Turing machine $M_L(a, x)$ for which

$$a \in L \Rightarrow \Pr[M_L(a,x) = 1] \geq 2/3$$

$$a \notin L \Rightarrow \Pr[M_L(a,x) = 0] \geq 2/3$$

where the probabilities are for an x picked uniformly in $\{0,1\}^{p(|x|)}$ for some polynomial p. If $M(a,x) = 1$, we say M accepts a using random tape x.

Let $L \in$ BPP, $a \in \{0,1\}^n$, and let $r' = p(n)$ be the number of random bits required above. The usual way of improving the success probability to $1 - 2^{-k}$ is to generate $O(k)$ purely random strings of length r', query a with these random strings, and take the majority vote. This requires $O(r'k)$ bits. We give a scheme where it suffices to use $O(k)$ pseudorandom strings of length r'; only $O(r' + k)$ random bits will be needed to generate this pseudo-random sequence.

To do this, we first use the usual technique to improve the success probability to .99. This requires $r = O(r')$ bits; we could also use the technique described later to use even fewer bits. Next, we take a random walk on an expander, an idea used in [AKS].

A random walk on an undirected graph G is the sequence of vertices $\{X_t\}$ visited by a particle that starts at a specified vertex and visits other vertices according to the following transition rule: if the particle is at vertex i at time t, then at time $t+1$ it moves to a neighbor of i picked uniformly at random. Associated with a random walk on G is the transition matrix A describing the probabilities of moving from one vertex to another. For convenience, we define A as the transpose of the traditional transition matrix, i.e. $A_{ij} = 1/\deg(j)$ if i and j are neighbors, and 0 otherwise.

The matrix A has eigenvalues $1 = \lambda_1 > \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_s \geq -1$. The only fact we'll need about an expander is that the associated transition matrix A has $\lambda_2 \leq 1 - \epsilon$, for some constant $\epsilon > 0$. We take G to be the 7-regular expander on the $s = 2^r$ vertices $\{0,1\}^r$ given in [GG]. This construction has the property that the neighbors of a vertex can be found in $O(1)$ time, so a random walk can be simulated efficiently.

It is also important for our construction that the λ_i are bounded away from -1. Unfortunately, G is bipartite, and hence $\lambda_s = -1$. To get around this, we do the following modified random walk. With probability $1/2$ we take a step in the random walk, and with probability $1/2$ we stay where we are. This modified walk has transition matrix $(I + A)/2$, which has eigenvalues $(1+\lambda_i)/2 \geq 0$. Thus, the second largest eigenvalue $1-\epsilon/2$ is still bounded away from 1.

We can now state our algorithm.

Algorithm: Let c be the least integer such that $(1 - \epsilon/2)^c \leq 1/10$. Use r random bits to find a uniformly random start vertex X_0, and perform the modified random walk $\{X_t\}$ for $t \leq 7ck$. For every $i \in \{1, 2, \ldots 7k\}$ query if $a \in L$ using the r-bit pseudo-random string represented by X_{ci}. Output the majority vote of these queries.

Proof of Correctness: We use the techniques in [AKS] to show that the error probability is at most 2^{-k}. First, some notation:

$C \subseteq G$ is the set of vertices representing strings for which queries are answered correctly.

P is the vector space R^s, where $s = 2^r$. P represents probabilities.

V is the subspace of multiples of $p_0 = (2^{-r}, 2^{-r}, \ldots, 2^{-r})$.

W is the subspace orthogonal to V.

For $p = (p_1, p_2, \ldots, p_s) \in P$

$|p| = \sum_{i=1}^{s} |p_i|$, the L_1-norm.

$||p|| = \sqrt{\sum_{i=1}^{s} p_i^2}$, the L_2-norm.

$B = \big((I + A)/2\big)^c$

$N : P \to P$ is the linear transformation

$$N(e_i) = \begin{cases} e_i & \text{if } i \in G - C \\ 0 & \text{otherwise,} \end{cases}$$

where e_i is the basis vector with a 1 in the ith place and 0's elsewhere.

$M = I - N$, where I is the identity.

Note that B is the transition matrix corresponding to c steps of the modified random walk. Thus, if the vector $p = (p_1, \ldots, p_s)$ represents the probabilities of being at the different vertices, i.e. $p_i = \Pr[\text{particle is at vertex } i]$, then Bp represents the probabilities after c steps in our modified random walk. Furthermore, NBp represents the probabilities of being at the different vertices and of not being in C. Extending this idea, we see that for a given sequence of the correctness of answers to queries, e.g. correct, correct, incorrect, ..., incorrect (denoted c,c,i,...,i)

$$\Pr[\text{c,c,i,...,i}] = |(NB)\ldots(NB)\cdots(NB)(MB)(MB)p_0|$$

To bound the right-hand side, it is easier to use L_2 norms. We need the following lemma:

Lemma 1 *For $p \in P$,*

(i) $||MBp|| \leq ||p||$.

(ii) $||NBp|| \leq ||p||/5$.

Proof.(i) follows from the definition of M and because the eigenvalues of B are between -1 and 1. To see (ii), write $p = v + w$, $v \in V$, $w \in W$. Using $Bv = v$ and $|G - C| \leq |G|/100$, we get $||NBv|| \leq ||v||/10 \leq ||p||/10$. The eigenvalues of B are $\big((1 + \lambda_i)/2\big)^c$, which by choice of c have absolute value at most 1/10, for $i \neq 1$. Since w is orthogonal to the eigenvector corresponding to λ_1, $||NBw|| \leq ||Bw|| \leq ||w||/10 \leq ||p||/10$. By the triangle inequality, $||NBp|| \leq ||NBv|| + ||NBw|| \leq ||p||/5$. ■

Using $|p| \leq 2^{s/2}||p||$, this lemma implies that for any sequence s with at least $7k/2$ incorrect answers,

$$\Pr[s] \leq 2^{s/2}||(NB)^{7k/2}p_0|| \leq 2^{s/2}(1/5)^{7k/2}||p_0|| = 5^{-7k/2}$$

Because there are at most 2^{7k} such sequences s,

$$\Pr[\exists s \text{ with} \geq 7\text{k}/2 \text{ incorrect answers}] \leq 2^{7k}5^{-7k/2} < 2^{-k},$$

which is what we wanted to show.

Remark. The above directly implies a version of Vazirani's result that quasi-random sources can be used to perform BPP computations[V2]. Specifically, there is a $\delta > -1$ such that if each r-bit string occurs with probability at most $2^{\delta r}$, the above construction works. To see this, say a prg needs Cr bits to reduce the probability of error to 2^{-r}. Then there are $2^{(C-1)r}$ "bad" strings of length Cr. Thus, if $\delta < (1 - C)/C$, the probability of error will be $2^{(C-1)r}2^{\delta rC} = 2^{-\Omega(r)}$.

3. Universal Hash Functions

Let H be a family of functions mapping $\{0,1\}^n$ to $\{0,1\}^l$. We say H is *universal* or a *universal family of hash functions* if, for every $x, y \in \{0,1\}^n, x \neq y$, the probability that $h(x) = h(y)$, for h selected randomly from H, is $1/2^l$ (see [CW]). We say H is *almost universal* if, for every such pair, the aforementioned probability is at most $1/2^l + 1/2^n$

One example of a universal family of hash functions is the set of all $n \times l$ matrices over the field with 2 elements. However, nl bits are required to specify an element of this family. The universal and almost universal families based on the shift register and linear congruential generators require only $O(n)$ bits, which will prove most useful.

Shift Register Hash Let $r = r_1 r_2 \ldots r_n, x = x_1 x_2 \ldots x_n \in \{0,1\}^n$. Let $r \cdot x$ denote the inner product of r and x modulo 2, i.e., $r \cdot x = 1$ iff the number of bit locations $i, 1 \leq i \leq n$ where $r_i = x_i = 1$, is odd. For $1 \leq l \leq n$, let $r^{(l)}$ denote the l'th shift of r, i.e. $r^{(l)} = r_{l+1} r_{l+2} \ldots r_n r_1 \ldots r_l$. The convolution of r on x, $r(x)$, is the string $r \cdot x, r^{(1)} \cdot x, \ldots, r^{(n-1)} \cdot x$.

Let p be a prime such that 2 is a generator modulo p. Let $n = p - 1$. Then let $C_{n,l}$ be the set of all strings of length p. Let the value of the function described by a string r on an n bit input x be the first l bits of $r(1 \circ x)$, where $\circ$ denotes concatenation. The results in [V] imply that $C_{n,l}$, interpreted as a family of functions as above, is universal. The length required to hash an n bit string using this method is the first prime $> n$ with 2 a generator. If the extended Riemann Hypothesis holds, this will be $O(n)$ (see [V]) and, for infinitely many n, will be $n + 1$. Since the prime involved is of size $O(n)$, rather than length $O(n)$, every number between n and cn can be tested until such a prime is found. For practical purposes, the time taken by this process is irrelevant, since p will be a fixed parameter.

Linear Congruential Hash mod p. Let p be a prime of length $n + 1$. Let $L_{n,l,p}$ be the set of pairs (a, b) of residues modulo p. We let (a, b) represent the function mapping input x to the l least significant bits of $ax + b$, where operations are performed modulo p. Now, $ax + b$ and $ay + b$ will be pairwise independent, i.e., for a fixed x and y, $ax + b$ and $ay + b$ are equally likely to be any pair of residues. However, their l least significant digits will not be pairwise independent, since the distribution on least significant bits of a random number modulo p is not uniform. However, no sequence of length l has probability exceeding $1/2^l + 1/p$; a simple calculation shows that this suffices for this family of hash functions to be almost universal.

Linear congruential hash functions as above require only $2n+2$ bits to represent. The disadvantage of this method is that it requires a prime of length $n+1$, which can be found quickly probabilistically, but not yet deterministically. In practice, this is not really a problem, since p is a fixed parameter. We could eliminate this technical obstacle by working in the field with 2^n elements instead, but calculations in an arbitrary field are messier than those modulo a prime. When converted into a prg, this method of hashing will be the closest to the methods in general use.

4. The Leftover Hash Lemma

We now show how to extract quasi-random strings from an unknown source of entropy. This is the key step in proving that the above families of hash functions yield quasi-perfect prg's.

First, we need some definitions from probability theory.

Definitions. Let D be a distribution on a finite set S. Denote by $D(s)$ for $s \in S$ the probability D assigns to s. For $X \subset S$, let $D(X)$ be the probability that an element chosen according to D is in X. Let the *collision probability* of D be the probability that two elements chosen independently according to D are the same. We say distributions D and D' are *statistically indistinguishable* within ϵ if, for every $X \subset S$, $|D(X) - D'(X)| < \epsilon$. We say D is *quasi-random* within ϵ on S if D is statistically indistinguishable from the uniform distribution on S within ϵ.

The following lemma is a somewhat stronger and cleaner version of a lemma in [ILL]; the strengthened version and proof found here are due to Rackoff [R].

Leftover Hash Lemma. Let $X \subset \{0,1\}^n, |X| \geq 2^l$. Let $e > 0$, and let H be an almost universal family of hash functions mapping n bits to $l-2e$ bits. Then the distribution $(h, h(x))$ is quasi-random within $1/2^e$ (on the set $H \times \{0,1\}^{l-2e}$), where h is chosen uniformly at random from H, and x uniformly from X.

We first show that the collision probability of the distribution $(h, h(x))$ is close to that of the uniform distribution on $H \times \{0,1\}^{l-2e}$. We then show that if a distribution on a set S has collision entropy close to $1/|S|$, then that distribution is quasi-random on S . More formally,

Claim 1. The collision probability of the distribution (h,h(x)) is at most $(1+2/2^{2e})/(|H|2^{l-2e})$.

Proof. The collision probability is the probability that for $h_1, h_2 \in H$, $x_1, x_2 \in X$ all randomly chosen, both $h_1 = h_2$ and $h_1(x_1) = h_2(x_2)$. This is $1/|H|$ times the probability that, for $x_1, x_2 \in X, h \in H$ randomly chosen, $h(x_1) = h(x_2)$. If $x_1 \neq x_2$, this probability is at most $1/2^{l-2e} + 1/2^n$ by the definition of almost universal. Since $|X| \geq 2^l$, the probability that $x_1 = x_2$ is at most $1/2^l$. Therefore, the collision probability is bounded by

$$1/|H| \cdot (1/2^{l-2e} + 1/2^n + 1/2^l) \leq (1+2/2^{2e})/(|H|2^{l-2e}).$$

Claim 2. Let D be a distribution on a finite set S. If the collision probability of D is at most $(1+2\delta^2)/|S|$, then D is quasi-random on S within δ.

Proof. By contradiction. Assume D were not. Then there exists an $X \subset S$ with $D[X] > |X|/|S| + \delta$. Without loss of generality, assume the above inequality is an equality. The collision probability of D is the probability that, for d_1, d_2 randomly chosen according to D, $d_1 = d_2$. The conditional probability that $d_1 = d_2$ given that both are in X is at least $1/|X|$; similarly, the conditional probability given that both are in $S - X$ is at least $1/(|S| - |X|)$. Thus, the collision probability is at least

$$\frac{D(X)^2}{|X|} + \frac{(1-D(X))^2}{|S|-|X|} = \frac{1}{|S|} + \frac{\delta^2}{|X|} + \frac{\delta^2}{|S|-|X|}$$

after substituting for $D(X)$ and simplifying. This last expression is minimized when $|X| = |S|/2$, so the collision probability is at least $(1+4\delta^2)/|S|$, contrary to assumption.

The Leftover Hash Lemma follows from combining Claims 1 and 2. ∎

5. PRG Based on Hash Functions

In this section, we apply the results of the previous section to show that any family of almost universal hash functions can be used to construct a quasi-perfect pseudorandom number generator. In particular, this yields modified versions of linear congruential and shift register generators which are provably quasi-perfect.

Construction of PRG. Let r, k, k' be integers. Let H be a family of almost universal hash functions mapping r bits to $r - k'$ bits. The pseudorandom generator takes as a seed an r bit string x, $h \in H$, and a sequence of $k-1$ strings of length k', $s_1, s_2, \ldots, s_{k-1}$. It outputs k r bit strings $x_1, \ldots, x_k$ defined as follows:

- $x_1 = x$.
- $x_{i+1} = h(x_i) \circ s_i$.

In other words, to get the next string in the sequence, hash the previous string, and add new random bits to replace the bits lost in hashing. In the case of the linear congruential hash, it is the same as first applying a linear congruential generator to x, then replacing the k' most significant bits of the result with new random bits, repeating this operation k times.

Theorem 2 *Let A be a BPP algorithm with error a constant $< 1/2$, using $r(n)$ random bits on an input of length n. Let k be any positive integer, $k' = 4k, r = r(n)$. Let A' be the following algorithm. A' first produces a random output $x_1, \ldots x_k$ of the Hash Based Generator with parameters r, k, and k'. A' simulates A on the input using each of the x_i and takes the majority output. Then A' has error probability $2^{-\Omega(k)}$.*

Proof. The idea is that the only information about x_1 that we use is the single bit, "Does A accept using random tape x_1?". Given this information, the conditional distribution of x_1 still has almost r bits of entropy. The Leftover Hash Lemma gives us a way of "recycling" almost all of this entropy for future use.

Without loss of generality, assume the input in question is in the language. Thus, the set of random tapes which cause A to accept has size at least $(2/3)2^r$. Let X be an arbitrary subset of the accepting strings of the aforementioned size, and let $b(z) = 1$ if $z \in X$, and 0 otherwise. We claim that for $1 \leq i \leq k$, the distribution $b(x_1), ..., b(x_i), h, x_{i+1}$ is statistically indistinguishable, within $2i/2^{2k}$, of the distribution consisting of i independent coin tosses of bias 2/3, followed by a random $h \in H$ and a random r bit string x. We prove this claim by induction on i.

For $i = 1$, this claim follows from the Leftover Hash Lemma: X and its complement have size at least $2^r/3$ so, for $j \in \{0, 1\}$, the distribution $(h, h(x_1))$ given that $b(x_1) = j$ is quasi-random within $2/2^{2k}$. The claim follows, since x_2 is simply $h(x_1)$ concatenated with truly random bits.

Assume the claim is true for $i - 1$. Then the distribution $b(x_1), ..., b(x_{i-1}), h, x_i$ cannot be distinguished from $i-1$ independent 2/3 biased coin flips, followed by a random h and x, with probability exceeding $2(i-1)/2^{2k}$. From the proof for $i = 1$, for h and x picked according to the latter distribution, $b(x), h, h(x)$ is be quasi-random within $2/2^{2k}$. Thus, a way of distinguishing $b(x_1), \ldots, b(x_i), h, x_{i+1}$ from a random sequence of the same form with probability $2i/2^{2k}$ would provide a way of distinguishing the distribution $b(x_1), ..b(x_{i-1}), h, x_i$ from a random sequence of the same form with probability $2(i-1)/2^{2k}$, contrary to the induction hypothesis.

Using this claim when $i = k$, we see that the sequence of $b(x_i)$'s is statistically indistinguishable within $2k/2^{2k}$ from k random independent coin flips of bias 2/3. Therefore, the probability that the majority of the $b(x_i)$ are 1 is $1 - 2^{-\Omega(k)}$, since this last is true for independent coin flips of bias 2/3. Since all elements of X cause A to accept, we are done.

Remark. Using either the Shift Hash or either Linear Congruential Hash, the above algorithm will use $O(r + k^2)$ random bits to obtain an error probability of 2^{-k}. In particular, for $k = r^{1/2}$, we obtain an error of $2^{-r^{1/2}}$ using $O(r)$ random bits. By "bootstrapping," we can reduce this error to $2^{-r^{1-\gamma}}$ using only $O(r)$ random bits.

6. Recycling Random Bits for Arbitrary Distributions

In this section, we generalize the technique of recycling random bits from "Yes" or "No" type algorithms to arbitrary distributions. This might be particularly useful for scientific simulations, where many bits are used, but only a few measurements of an experiment are recorded. The methods presented here could be used to recycle these bits for use in future experiments, or even for later stages of the same experiment. Other situations where these techniques might be used are: cryptography, where it might be necessary to generate many keys or problem instances from a certain distribution, such as a long list of primes, or pre-factored Blum integers; approximation algorithms such as in [JVV], where many samples from a distribution are used to determine a succinctly describable quantity; or any probabilistic algorithm which has a subroutine or subroutines which output fewer bits than the amount of randomness they use.

The methods presented here can also be used to generate a random prime, for example, using few random bits. This will appear in the final version.

We now define the entropy of a distribution.

Definition. Let D be a distribution on a finite set S. Then the entropy of D is given by

$$\sum_{s \in S} -D(s)\log(D(s)) = E_D[-\log(D(s))]$$

where $E_D(\cdot)$ denotes the expectation of $(\cdot)$ with s chosen according to D.

Intuitively, the entropy of a distribution measures the amount of randomness in the distribution. We talk of entropy as being measured in "bits". The uniform distribution on a set S has entropy $\log(|S|)$.

Theorem 3 *Let $D(z)$ be an indexed family of distributions so that $D(z)$ can be sampled from in polynomial-time using $r(|z|)$ random bits. Let $E(z)$ be a polynomial-time function so that $E(z)$ is always larger than the entropy of $D(z)$. Then t samples from $D(z)$ can be made in polynomial time using $E(z)t + O(t^{5/6}r(|z|))$ random bits, so that the sequence output is statistically indistinguishable from a random such sequence within an exponentially small (in t) error.*

Note. If $E(z)$ gives a close bound on the entropy, and the entropy is always non-negligible (say, at least 1/poly(n)), then, for sufficiently large $t(n)$, the amortized random bits per sample is the entropy $+o(1)$. This is the information-theoretic lower bound. Thus, the above theorem can be thought of as an effective version of Shannon's Theorem [Sh].

We require some lemmas from probability theory. The following lemma follows from the Martingale Tail Inequality [Sp].

Lemma 4 *Let D be a distribution on the interval $[a, b]$, with expectation E. Let t be a positive integer, and let $0 < \alpha < t^{1/2}$. Pick $d_1, \ldots, d_t$ independently according to D. Then*

$$Pr[|\sum_{1 \leq i \leq t} d_i - tE| > \alpha t^{1/2}(b-a)] < 2e^{-\alpha^2/2}$$

Corollary 5 *Let f be a function on $\{0,1\}^r$. Let E be the entropy of the distribution $f(x)$ for x chosen randomly. Let f^t be the function mapping the tr bit string $x_1 \circ \ldots \circ x_t$ to $f(x_1) \circ \ldots \circ f(x_t)$. Then, for $x = x_1 \circ \ldots \circ x_t$ chosen at random, with probability at least $1 - 2e^{-\alpha^2/2}$, $f^t(x)$ has at least $2^{(r-E)k - r\alpha t^{1/2}}$ pre-images under f^t.*

Proof. Apply the previous lemma to the distribution $D = log|f^{-1}(f(x))|$. D is a distribution on $[0, r]$ with mean $r - E$. The log of the number of preimages of x under f^t has the same distribution as the sum of t random elements of D.

Proof of Theorem 3. For input parameter z, let $r = r(|z|), E = E(z), D = D(z)$, and let T be the number of samples desired. Let $R = rT^{2/3}, k = T^{1/3}$, and $k' = ET^{2/3} + 2rT^{1/2}$. Use the Hash Based prg, with parameters R, k, and k', to generate k pseudo-random numbers $x_1, \ldots, x_k$ of length R (using $O(R + kk') = O(ET + rT^{5/6})$ bits.) Let A be the algorithm which takes an R bit number, divides it into $T^{2/3}$ blocks of size r, and uses each block to simulate the sampling algorithm. Output $A(x_1), \ldots, A(x_k)$, a total of T outputs. We claim this output is statistically indistinguishable from T independent samples from D.

Analogously to Theorem 2, we prove, by induction on i, that the distribution $A(x_1)...A(x_i), h, x_{i+1}$ is indistinguishable from the distribution $d_1, ..d_i, h, x'$, (i random samples from D followed by a random $h \in H$ and R bit string x') within $4ie^{-T^{1/3}/2}$. We prove the case $i = 1$; the induction step is exactly as in Theorem 4.1.

Let $E' \leq E$ be the entropy of D. Note that, by Corollary 5, for x a random r bit string, $A(x)$ has at least $P = 2^{(r-E')T^{2/3} - rT^{1/2}}$ pre-images with probability at least $1 - 2e^{-T^{1/3}/2}$. (Here, we use $t = T^{2/3}, \alpha = T^{1/6}$). Assume $y = A(x)$ has at least P pre-images under A; let X be the set of pre-images of y. $|X| \geq P \geq 2^{R-k'+rT^{1/2}}$, so applying the Leftover Hash Lemma to X, we have that the distribution h, x_2 given that x_1 is in X is quasi-random within $2^{-rT^{1/2}/2}$. Since this holds for all such y, we conclude that the distribution $A(x_1), h, x_2$ given that $A(x_1)$ has at least P pre-images is statistically indistinguishable from $A(x_1), h, x'$ given that $A(x_1)$ has at least P preimages. Since this event occurs with exponentially high probability, we are done. $\blacksquare$

An even more general theorem along these lines can be proved.

Theorem 6 *Let a probabilistic task on r bits mean a circuit C with r inputs, and a number $E > 1$ which is at least the entropy of the distribution D_C given by $C(x)$ for x chosen randomly. To fulfill a task means to produce an element according to D_C. A request distribution is a distribution on sequences of tasks, so that the task requested at a given time depends only on the outputs of the previous tasks (when fulfilled). Then there is a polynomial-time algorithm which fulfills tasks on-line using, in the limit, $o(t) + \sum_{1 \leq i \leq t} E_i$ random bits to fulfill the first t tasks to within an exponentially small statistical error, when run on any request distribution for a polynomial number of requests.*

The proof is similar to that of the last theorem.

7. Acknowledgements

We thank Charlie Rackoff for his proof of the Leftover Hash Lemma, and Umesh Vazirani for much technical and philosophical advice through the course of our research. We also thank Leonid Levin, Mike Luby, Steven Rudich, Moni Naor, Noam Nisan, and Mauricio Karchmer for helpful discussions.

8. References

[AKS] M. Ajtai, J. Komlos, and E. Szemeredi, "Deterministic Simulation of Logspace," 19th STOC, 1987.

[Ba] E. Bach, "Realistic Analysis of Some Randomized Algorithms", 19th STOC, 1987.

[BM] M. Blum and S. Micali, "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits," SIAM J. Comput., 13:270-299 (1984).

[CW] L. Carter and M. Wegman, "Universal Hash Functions," JCSS, 1979.

[CWi] A. Cohen and A. Wigderson, "Dispersers, Deterministic Amplification, and Weak Random Sources," 30th FOCS, 1989.

[FKL] D. Frieze, R. Kannan, and J. C. Lagarias, "Linear Congruential Generators Do Not Produce Random Sequences," 25th FOCS, 1984.

[GG] O. Gabber and Z. Galil, "Explicit Construction of Linear-Sized Superconcentrators," 20th FOCS, 1979.

[ILL] R. Impagliazzo, L. Levin, and M. Luby, "Pseudo-random generation from one-way functions," 21st STOC, 1989.

[JVV] M. Jerrum, L. Valiant, and V. Vazirani "Random Generation of Combinatorial Structures from a Uniform Distribution," Theoretical Computer Science, 43:169-188 (1986).

[KR] H. Karloff and P. Raghavan, "Randomized Algorithms and Pseudorandom Numbers," 20th STOC, 1988.

[LPS] A. Lubotzky, R. Phillips, and N. Sarnak "Explicit Expanders and the Ramanujan conjecture," 18th STOC, 1986.

[P] J. Plumstead, "Inferring a Sequence Generated by a Linear Congruence," 24th FOCS, 1983.

[R] M. O. Rabin, "Probabilistic algorithm for testing primality," J. of Number Theory, 12:128-138 (1980).

[Sa] M. Santha, "On Using Deterministic Functions to Reduce Randomness in Probabilistic Algorithms," Manuscript.

[Sh] C.E. Shannon, "A Mathematical Theory of Communication," Bell Syst. Tech. J., 27:379-423,623-656 (1948).

[Si] M. Sipser, "Expanders, Randomness, or Time versus Space," Structure in Complexity, 1986.

[Sp] J. Spencer, "Ten Lectures on the Probabilistic Method," SIAM, Philadelphia, 1987, pp. 55-56.

[V] U. Vazirani, "Efficiency Considerations in Using Semi-Random Sources," 19th STOC, 1987.

[V2] U. Vazirani, "Randomness, Adversaries and Computation," PhD Thesis, University of California, Berkeley, 1986.

[Y] A. Yao, "Theory and Applications of Trapdoor Functions," 23rd FOCS, 1982.