ON THE CONSTRUCTION OF RANDOM NUMBER GENERATORS AND RANDOM FUNCTION GENERATORS

C. P. Schnorr
Universität Frankfurt
Fachbereich Mathematik/Informatik
6000 Frankfurt, West Germany

Abstract. Blum, Micali (1982), Yao (1982), Goldreich, Goldwasser and Micali (1984), and Luby, Rackoff (1986) have constructed random number generators, random function generators and random permutation generators that are perfect if certain complexity assumptions hold. We propose random number generators that pass all statistical tests that depend on a small fraction of the bitstring. This does not rely on any unproven hypothesis. We propose improved random function generators with short function names and which minimize the number of pseudo-random bits that are necessary for the evaluation of pseudo-random functions. We announce a new very efficient perfect random number generator.

1. Random generators without unproven assumptions

Let $I_n = \{0,1\}^n$, $H_n = I_n^{I_n}$ = "the set of all functions $f : I_n \to I_n$". A *random function* generator is an efficient algorithm F that generates from names $x \in I_m$ a function $F_{m,x} \in H_{k(m)}$ for some function k(m); when given for input m,x,y the algorithm computes $F_{m,x}(y)$. We associate with $f \in H_n$ a function $F_{n,f} \in H_{2n}$ defined by

$$F_{n,f}(l,r) = (r, l \oplus f(r)) \qquad \text{for all } l,r \in I_n. \tag{1}$$

The function $F_{n,f}$ roughly corresponds to a layer in the DES-algorithm. We consider $F_{n,f}^{(3)} = F_{n,f}\, F_{n,f}\, F_{n,f}$ as a random function generator for the functions $F_{n,f}^{(3)}$ in H_{2n} and with names $f \in H_n$. The functions $F_{n,f}^{(3)}$ are permutations, and $F_{n,f}^{(3)}$ is called a *random permutation generator*. Luby and Rackoff have considered the random function generator $F_{n,f_3}\, F_{n,f_2}\, F_{n,f_1}$ where independent random functions $f_1,f_2,f_3 \in H_n$ are used at each stage. We observe that the analysis of Luby and Rackoff remains valid for the case that $f_1 = f_2 = f_3$. This yields the following version of the main theorem in Luby, Rackoff (1986).

Theorem 1. (Luby, Rackoff (1986)) *For random $f \in H_n$ the function $F_{n,f}^{(3)} = F_{n,f}\, F_{n,f}$*

C.G. Guenther (Ed.): Advances in Cryptology - EUROCRYPT '88, LNCS 330, pp. 225-232, 1988.

$F_{n,t}$ passes all statistical function tests that are restricted to $2^{o(n)}$ oracle queries.

The concept of *statistical function test* has been introduced by Goldreich, Goldwasser, Micali (1984). A test T is a probabilistic algorithm with 0,1-output, which is endowed with an oracle O_g for evaluating the function g at inputs y computed by the test T; the value g(y) is computed by a single step using the oracle. One associates the following probabilities to a statistical test T and a random function generator F. Let p_n^H (p_n^F, resp.) be the probability that T with oracle O_g gives output 1 when $g \in H_n$ is chosen at random with uniform probability ($g \in H_n$ is chosen at random from F, resp.). The probability space is the set of all internal coin tosses of T and of all choices for g. In the proof of Theorem 1 Luby and Rackoff have shown that the above generator $F_{n,t}^{(3)}$ satisfies

$$|p_n^H - p_n^F| = O(m^3 2^{-n})$$

for every statistical function test T that is limited to at most m oracle queries.

One defines that a function generator F *passes* the function test T if

$$|p_n^H - p_n^F| = O(n^{-t}) \quad \text{for all } t > 0 .$$

A random function generator is called *perfect* if it passes all statistical function tests with polynomial time bound $n^{O(1)}$. The functions generated by a perfect random function generator are called *pseudo-random*.

Theorem 1 is strong in the sense that there is no time bound for the statistical tests and the bound $2^{o(n)}$ on the number of oracle queries is superpolynomial in n. On the other hand the *name* $f \in H_n$ for the function $F_{n,t}^{(3)} \in H_{2n}$ is $n2^n$ bits long whereas Goldreich, Goldwasser, Micali (1984) construct pseudo-random functions in H_n with names in I_n. The proof of Theorem 1 follows from the analysis of the Luby, Rackoff (1986) random permutation generator. The technical proof is quite involved.

A *random number generator* is an efficient algorithm which transforms short random seeds into long pseudo-random strings. Every random function generator gives rise to a corresponding random number generator and vice-versa. There is a natural bijection $\Phi_n : H_n \to I_{n2^n}$ which maps functions $f \in H_n$ into the concatenation $\Phi_n(f) = \prod_{x \in I_n} f(x)$ where x ranges over all strings $x \in I_n$ in alphabetical order. By this bijection the above function $F_{n,t}^{(3)}$ yields a function

$$\begin{aligned} G_n : I_{n2^n} &\to I_{2n2^{2n}} \\ x &\mapsto \Phi_{2n}\left[F_{n,\Phi_n^{-1}(x)}^{(3)}\right] . \end{aligned}$$

We give a more concrete description of the random number generator

$$G_n : I_{n2^n} \to I_{2n2^{2n}} , \quad G : x \mapsto y .$$

We write the input string $x \in I_{n2^n}$ as concatenation of 2^n strings in I_n, and we enumerate these 2^n substrings of x using indices in I_n:

$$I_{n2^n} \ni \quad x = \prod_{i \in I_n} x_i \quad \in (I_n)^{2^n} .$$

We likewise partition the output string $y \in I_{2n2^{2n}}$:

$$I_{2n2^{2n}} \ni \quad y = \prod_{i \in I_{2n}} y_i \quad \in (I_{2n})^{2^{2n}} .$$

For every string $y \in I_{2n}$ let $L(y)$, $R(y)$ be the left and right half string in I_n:

$$I_{2n} \ni \quad y = L(y)\, R(y) \quad \in (I_n)^2 .$$

Algorithm for G_n

input $\quad x = \prod_{i \in I_n} x_i$.

1. $y_i^0 := i \qquad$ for all $i \in I_{2n}$.
2. For $j = 0,1,2$ do
   $y_i^{j+1} := R(y_i^j)\, (L(y_i^j) \oplus x_{R(y_i^j)})$.

output $\quad y^3 = \prod_{i \in I_{2n}} y_i^3$.

Each iteration step switches the left and right part of $y \in I_{2n}$ and adds to the new right part the substring $x_{R(y)}$ of the input x; here $\oplus$ is the vector addition modulo 2. According to the bijections Φ_n, Φ_{2n} Theorem 1 translates into Theorem 2.

Theorem 2. *The random number generator* $(G_n)_{n \in \mathbb{N}}$, $G_n : I_{n2^n} \to I_{2n2^{2n}}$, *passes all statistical number tests that depend on at most* $2^{o(n)}$ *bits of* $G_n(x)$.

A *statistical number test* T is a probabilistic algorithm which takes for input a binary string, and gives a 0,1-output (Yao, 1982). One associates with T and a random number generator G the following probabilities. Let p_k^I (p_k^G, resp.) be the probability that T outputs 1 when given for input a random string $x \in I_k$ with uniform distribution (a string $y \in I_k$ chosen at random from G, resp.). The number generator G *passes* the test if

$$|p_k^I - p_k^G| = O(k^{-t}) \quad \text{for all } t > 0 .$$

A random number generator is called *perfect* if it passes all polynomial time statistical number tests. The bit strings generated by a perfect random number generator are called *pseudo-random*.

Theorem 2 means that every selection of at most $m = 2^{o(n)}$ bits from $G_n(x)$ passes all statistical number tests T (even tests with arbitrary time bounds) provided that $x \in I_{n2^n}$ is random with uniform probability. The bit strings $G_n(x)$ are, for random seed $x \in I_{n2^n}$, completely randomized locally. Every statistical number test that distinguishes the distribution of $G_n(x) \in I_{2n2^{2n}}$ from the uniform distribution on

$I_{2n2}{}^{2n}$ depends on at least a polynomial fraction of the bit string $G_n(x)$.

So far we have seen that the above number generator G_n is based on a powerful construction principle for local randomization. It is an important question whether this construction principle also yields good global random properties. We next prove that all strings that are locally randomized satisfy the law of large numbers.

Theorem 3 *Let $(\overline{G}_n)_{n\in\mathbb{N}}$ be a random number generator $\overline{G}_n : I_n \to I_2{}^n$ such that $\overline{G}_n(x)$, for random $x \in I_n$ passes all statistical test that depend on at most $2^{o(n)}$ bits of $G_n(x)$. Then the frequency of ones and zeroes in $\overline{G}_n(x)$ is approximately 1/2.*

Proof. Consider the statistical test that selects $m = 2^{o(n)}$ independent random bits $y_1,\dots,y_m$ from the bit string $\overline{G}_n(x)$ and computes $\#_1(y)$ = "the number of ones in these bits". These bit strings y pass all statistical tests. By Chebyshev's inequality this implies

$$\text{prob}\left[\,\left|\#_1(y)/m - \frac{1}{2}\right| \geq \varepsilon\right] \;\leq\; 1/(\varepsilon^2 m) + O(m^{-t}) \quad \text{for all } \varepsilon > 0 \text{ and all } t > 0\,.$$

The probability space is the set of all seeds $x \in I_n$ and of all possible selections of substrings y. Note that the expected value of $\#_1(y)/m$ and of $\#_1(\overline{G}_n(x))/2^n$ coincide. Therefore we obtain for $\varepsilon = (1/m)^{1/3}$ and $m = 2^{n/\log n}$

$$\text{prob}\left[\,\left|\#_1(\overline{G}_n(x))/2^n - \frac{1}{2}\right| \geq 2^{-n/(3\log n)}\,\right] \;\leq\; 2^{-n/(3\log n)} + O(2^{-n/\log n})\,.$$

We next show that the upper bound $2^{o(n)}$, limiting the number of oracle queries, in Theorem 1 is sharp. We associate to $f \in H_n$ the function generator

$$F_{n,f}^{(\nu)} = F_{n,f}\, F_{n,f} \cdots F_{n,f} \quad \nu\text{-times.}$$

Theorem 4. *There is a statistical function test that rejects the function generators $F_{n,f}^{(\nu)}$ for all $\nu \in \mathbb{N}$, using $O(2^n)$ oracle queries.*

Proof. We have for all $r, l \in I_n$:

$$F_{n,f}(l,r) = (r, l \oplus f(r))$$
$$F_{n,f}^{-1}(l,r) = (r \oplus f(l), l)\,.$$

This implies that for all $\nu \geq 1$

$$F_{n,f}^{(\nu)}(l,r) = F_{n,f}^{(\nu+1)}(r \oplus f(l),\, l)\,,$$

and thus

$$L\, F_{n,f}^{(\nu)}(l,r) = R\, F_{n,f}^{(\nu)}(r \oplus f(l),\, l)\,. \qquad (2)$$

A statistical test for verifying the relation (2) fixes r and l and tries for $f(l) \in I_n$ all bit strings $y \in I_n$. Once f(l) has been found the relation (2) holds for all r. The

statistical test requires at most $O(2^n)$ oracle queries in order to find f(1); it evaluates $F^{(\nu)}_{n,f}(1,r)$ and $F^{(\nu)}_{n,f}(r \oplus y, 1)$ for all strings $y \in I_n$. □

The above statistical test does not reject function generators

$$F_{n,f_3} F_{n,f_2} F_{n,f_1}$$

where distinct functions f_1, f_2, f_3 are used at each stage.

2. Improved random function generators

Goldreich, Goldwasser and Micali (1984) show that every perfect random number generator $(\overline{G}_n)_{n\in\mathbb{N}}$, $\overline{G}_n : I_n \to I_{2n}$, can be transformed into a perfect random function generator $(\overline{F}_n)_{n\in\mathbb{N}}$, $\overline{F}_{n,x} \in H_n$ with $x \in I_n$, such that functions $\overline{F}_{n,x} \in H_n$ have names x of length n and can be evaluated using $O(n^2)$ pseudo-random bits generated by $\overline{G}_n$. We improve this construction via the Luby, Rackoff permutation generator.

Theorem 5. *For every $\varepsilon > 0$ every perfect random number generator $(\overline{G}_n)_{n\in\mathbb{N}}$, with $\overline{G}_n : I_n \to I_{2n}$, can be transformed into a perfect random function generator $(\overline{F}_n)_{n\in\mathbb{N}}$ such that*

(1) *$\overline{F}_{n,x} \in H_n$ has names x of length $(\log n)^{1+\varepsilon}$.*

(2) *evaluation of $\overline{F}_{n,x}$ can be done using $O(n(\log n)^{1+\varepsilon})$ pseudo-random bits generated from $\overline{G}_n$.*

Sketch of proof. By the construction of Goldreich, Goldwasser, Micali (1984) we generate, from pseudo-random bits obtained from $\overline{G}_n(x)$, a pseudo-random function $f \in H_{m(\varepsilon)}$, $m(\varepsilon) = (\log n)^{1+\varepsilon}$, that passes all function tests with time bound $n^{O(1)}$. These functions $f \in H_{m(\varepsilon)}$ have names in $I_{m(\varepsilon)}$ and can be evaluated using $(\log n)^{2+2\varepsilon}$ pseudo-random bits. It follows from Theorem 1 and since $n^t = 2^{o((\log n)^{1+\varepsilon})}$ for all $t > 0$ and all $\varepsilon > 0$, that the functions $F^{(3)}_{m(\varepsilon),f} \in H_{2m(\varepsilon)}$ pass all statistical function tests that have time bound $n^{O(1)}$.

In a way similar to (1) we associate with $f \in H_{m(\varepsilon)}$ a function $\tilde{F}_{n,f} \in H_n$ defined by

$$\tilde{F}_{n,f}(B_1,\ldots,B_k) = (B_2,\ldots,B_k\ ,\ B_1 \oplus f(B_k)) \qquad (3)$$

for all $B_1,\ldots,B_k \in I_{m(\varepsilon)}$ with $k = n/m(\varepsilon)$. By the same argument that proves Theorem 1, we can show that

$$\overline{F}_{n,x} := \tilde{F}^{(2k-1)}_{n,f} \in H_n$$

passes all statistical function tests with time bound $n^{O(1)}$. □

3. New efficient and perfect pseudo-random number generators

S. Micali and C.P. Schnorr (1988) introduce new random number generators (RNG) that are perfect under a reasonable complexity assumption and that are nearly as efficient as the popular linear congruential generator which is known to be imperfect.

A RNG is *perfect* if it passes all polynomial time statistical tests, i.e. the distribution of output sequences cannot be distinguished, by probabilistic polynomial time algorithms, from the uniform distribution of sequences of the same length. So far the proofs of perfectness are all based on unproven complexity assumptions. This is because we cannot prove superpolynomial complexity lower bounds.

Perfect random number generators have been established for example based on the discrete logarithm by Blum, Micali (1982), based on quadratic residuosity by Blum, Blum, Shub (1986), based on one way functions by Yao (1982), based on RSA encryption and factoring by Alexi, Chor, Goldreich and Schnorr (1984). All these RNG's are less efficient than the linear congruential generator. The RSA/RABIN-generator is the most efficient of these generators. It successively generates log n pseudo-random bits by one modular multiplication with a modulus N that is n bits long.

The RSA-generator can be extended and accelerated in various ways. A new powerful complexity assumptions yields more efficient generators. Let $N = pq$ be product of two large random primes p and q and let d be a natural number that is relatively prime to $\varphi(N) = (p-1)(q-1)$. It is conjectured that the following distributions are indistinguishable by efficient statistical tests:

- the distribution of $x^d \pmod N$ for random $x \in [1,N^{2/d}]$.
- the uniform distribution on $[1,N]$.

This hypothesis is closely related to the security of the RSA-scheme. Under this hypothesis the transformation

$$[1,N^{2/d}] \ni x \mapsto x^d \pmod N \in [1,N]$$

stretches short random seeds $x \in [1,N^{2/d}]$ into pseudo-random numbers $x^d \pmod N$ in the interval [1,N]. Various random number generators can be built on this transformation. The sequential polynomial generator generates from random seed $x \in [1,N^{2/d}]$ a sequence of numbers $x = x_1,x_2,...,x_i,... \in [1,N^{2/d}]$. The n(1-2/d) least significant bits of the binary representation of $x_i^d \pmod N$ are the output of x_i and the 2n/d most significant bits form the successor x_{i+1} of x_i.

It follows from a general argument of Goldreich, Goldwasser, Micali (1984) and the above hypothesis that all these generators are perfect, i.e. the distribution of output strings is indistinguishable, by polynomial time statistical tests, from the uniform distribution of binary strings of the same length. The sequential generator is nearly as efficient as the linear congruential generator. Using a modulus N, that is n bit long, it outputs n(1-2/d) pseudo-random bits per iteration step. The costs of an iteration step $x \mapsto x^d \pmod N$ with $x \in [1,N^{2/d}]$ corresponds to the costs of about one full multiplication modulo N. This is because the evaluation of $x^d \pmod N$ over numbers $x \leq N^{2/d}$ consists almost entirely of multiplications with small numbers that do not require modular reduction.

Micali and Schnorr extend the sequential polynomial generator to a parallel polynomial generator (PPG). The PPG generates from random seed $x \in [1,N^{2/d}]$ a tree. The nodes of this iteration tree are pseudo-random numbers in $[1,N^{2/d}]$ with outdegree at most d/2. To compute the successor nodes y(1),...,y(s) and the output string of node y one stretches y into a pseudo-random number $y^d \pmod N$ that is n bits long. Then the successors y(1),...,y(s) of y are obtained by partitioning the most significant bits of $y^d \pmod N$ into $s \leq d/2$ bit strings of length $\lfloor 2n/d \rfloor$. The output of node y consists of the remaining least significant bits of $y^d \pmod N$. Any collection of subtrees of the iteration tree can be independently processed in parallel once the corresponding roots are given. In this way m parallel processors can speed the generation of pseudo-random bits by a factor m. These parallel processors need not to communicate; they are given pseudo-independent input strings and their output strings are simply concatenated. The concatenated output of all nodes of the iteration tree is pseudo-random, i.e. the parallel generator is perfect. The PPG enables fast retrieval of substrings of the pseudo-random output. To access a node of the iteration tree we follow the path from the root to this node. After retrieving a bit the subsequent bits in the output can be generated at full speed. Iteration trees of depth at most 60 are sufficient for practical purposes; they generate pseudo-random strings of length 10^{20} (for outdegree 2) such that individual bits can be retrieved within a few seconds.

The parallel generator is based on a method that has been invented by Goldreich,

Goldwasser and Micali (1984) for the construction of random functions. Micali and Schnorr observe that this construction can be applied to speed every perfect random number generator by a factor m using m parallel processors. Using this principle and sufficiently many parallel processors we can generate pseudo-random bits with almost any speed. This important method of parallelization applies to all perfect random number generators but the RSA-generator is particularly suited for this method. The method of parallelization does not apply to imperfect random number generators like the linear congruential generator since this method can further detoriate a weak generator.

References

Alexi, W., Chor, B., Goldreich, O., and Schnorr, C.P.: RSA and Rabin Functions: certain parts are as hard as the whole. Proceeding of the 25th Symposium on Foundations of Computer Science, 1984, pp. 449-457; also: Siam Journal on Comput., (1988).

Blum, L., Blum, M. and Shub, M.: A simple unpredictable pseudo-random number generator. Siam J. on Computing (1986), pp. 364-383.

Blum, M. and Micali, S.: How to generate cryptographically strong sequences of pseudo-random bits. Proceedings of the 25th IEEE Symposium on Foundations of Computer Science, IEEE, New York (1982); also Siam J. Comput. 13 (1984) pp. 850-864.

Goldreich, O., Goldwasser, S., Micali, S.: How to Construct Random Functions. Proceedings of the 25th IEEE Symposium on Foundations of Computer Science, IEEE, New York, (1984); also Journal ACM 33,4 (1986) pp. 792-807.

Luby, M. and Rackoff, Ch.: Pseudo-random permutation generators and cryptographic composition. Proceedings of the 18th ACM Symposium on the Theory of Computing, ACM, New York (1986) pp. 356-363.

Micali, S. and Schnorr, C.P.: Efficient, perfect random number generators. preprint MIT, Universität Frankfurt 1988.

Yao, A.C.: Theory and applications of trapdoor functions. Proceedings of the 25th IEEE Symposium on Foundations of Computer Science, IEEE, New York (1982), pp. 80-91.