Statistical Science
1993, Vol. 8, No. 1, 31–39

Pseudorandom Numbers

Jeffrey C. Lagarias

Abstract. This article surveys the problem of generating pseudorandom numbers and lists many of the known constructions of pseudorandom bits. It outlines the subject of computational information theory. In this theory the fundamental object is a secure pseudorandom bit generator. Such generators are not theoretically proved to exist, although functions are known that appear to possess the required properties. In any case, pseudorandom number generators are known that work reasonably well in practice.

Key words and phrases: Computational information theory, cryptography, data encryption standard, discrete exponentiation, multiplicative congruential generator, one-way function, private key cryptosystem, pseudorandom numbers, RSA public key cryptosystem.

1. INTRODUCTION

A basic ingredient needed in any algorithm using randomization is a source of "random" bits. In practice, in probabilistic algorithms or Monte Carlo simulations, one uses instead "random-looking" bits. A *pseudorandom bit generator* is a deterministic method to produce from a small set of "random" bits called the *seed* a larger set of random-looking bits called pseudorandom bits.

There are several reasons for using pseudorandom bits. First, truly random bits are hard to come by. Physical sources of supposedly random bits, which rely on chaotic, dissipative processes such as varactor diodes, typically produce correlated bits rather than independent sequences of bits. Such sources also produce bits rather slowly (Schuster, 1988). Hence one would like to conserve the number of random bits needed in a computation. Second, the deterministic character of pseudorandom bit sequences permits the easy reproducibility of computations. A third reason arises from cryptography: the existence of secure pseudorandom bit generators is essentially equivalent to the existence of secure private-key cryptosystems.

For Monte Carlo simulations, one often wants *pseudorandom numbers*, which are numbers simulating either independent draws from a fixed probability distribution on the real line $\mathbf{R}$ or more generally numbers simulating samples from a stationary random process. It is possible to simulate samples of any reasonable distribution using as input a sequence of i.i.d. 0–1 valued random variables (Devroye, 1986; and Knuth and Yao, 1976). Hence the problem of constructing pseudorandom numbers is in principle reducible to that of constructing pseudorandom bits.

Section 2 describes explicitly some pseudorandom bit generators, as well as some general principles for constructing pseudorandom bit generators. Most of these generators have underlying group-theoretic or number-theoretic structure.

Section 3 describes the subject of computational information theory and indicates its connection to cryptography. The basic object in this theory is the concept of a secure pseudorandom bit generator, which was proposed by Blum and Micali (1984) and Yao (1982). The basic properties characterizing a secure pseudorandom bit generator are "randomness-increasing" and "computationally unpredictable." Recently obtained results are that if one of the following objects exists, then they all exist:

1. a secure pseudorandom bit generator;
2. a one-way function; and
3. a secure (block-type) private-key cryptosystem.

The central unsolved question is whether any of these objects exist. However, good candidates are known for one-way functions. A major difficulty in settling the existence problem for this theory is summarized in the following heuristic.

UNPREDICTABILITY PARADOX. If a deterministic function is unpredictable, then it is difficult to prove anything about it; in particular, it is difficult to prove that it is unpredictable.

Jeffrey C. Lagarias is a Distinguished Member of Technical Staff, Math Sciences Research Center, AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, New Jersey 07974.

Section 4 surveys results on the statistical performance of various known pseudorandom number generators. In particular, a pseudorandom bit generator called the RSA generator produces secure pseudorandom bits (in the sense of Section 2) if the problem of factoring certain integers is computationally difficult. A good general reference for pseudorandom number generators is Knuth (1981, Chapter 3).

2. EXPLICIT CONSTRUCTIONS OF PSEUDORANDOM BIT GENERATORS

Where might pseudorandom number generators come from? There are several general principles useful in constructing deterministic sequences exhibiting quasi-random behavior: expansiveness, nonlinearity and computational complexity.

Expansiveness is a concept arising from dynamical systems. A flow f_t on a metric space is *expansive* (or *hyperbolic*) at a point $x \in M$ if nearby trajectories diverge (locally) at an exponential rate from each other, that is, $|f_t(x) - f_t(y)| \geq \lambda^t |x - y|$, where $\lambda > 1$, provided $|x - y| < \delta$, and for $0 \leq t < t_0$. Such flows exhibit sensitive dependence on initial conditions and are numerically unstable to compute. A discrete version of this phenomenon for maps on the interval $I = [0, 1]$ is a map $T: I \to I$ such that $|T'(x)| > 1$ for almost all $x \in [0, 1]$; for example, $T(x) = \theta x \pmod 1$, where $\theta > 1$. In particular, it requires knowledge of at least the first $O(n)$ bits of $T^{(n)}(x)$ to determine if $x \in [0, 1/2)$ or $x \in [1/2, 1)$ as $n \to \infty$.

Nonlinearity provides a second source of deterministic randomness. Functional compositions of linear maps are themselves linear, but nonlinear polynomial maps, when composed, can yield exponentially growing nonlinearity; for example, if $f(x) = x^2$ and $f^{(j)}(x) = f(f^{(j-1)}(x))$, then $f^{(n)}(x) = x^{2^n}$. Even the simple nonlinear map $f(x) = ax(1 - x)$ for $a \in [0, 4]$, which maps $[0, 1]$ into $[0, 1]$, exhibits extremely complicated dynamics under iteration (Collet and Eckmann, 1980).

Computational complexity is a third source of deterministic randomness. Kolmogorov (1965), Chaitin (1966) and Martin-Lof (1966) defined a finite string $A = a_1 \cdots a_k$ of bits to be *Kolmogorov-random* if the length of the shortest input to a fixed universal Turing machine $\mathfrak{I}$ that halts and outputs A is of length greater than or equal to $k - c_0$ for a constant c_0. This notion of randomness is that of *computational incompressibility*. Unfortunately, it is an undecidable problem to tell whether a given string A is Kolmogorov-random. However, one obtains weaker notions of computational incompressibility by restricting the amount of computation allowed in attempting to compress the string A. For example, given a nondecreasing time-counting function T such as $T(x) = 5x^3$, a string A is *T-incompressible* if the shortest input to $\mathfrak{I}$ that halts in $T(|A|)$ steps and outputs A has length greater than $|A|$. This notion of incompressibility is effectively computable. One can similarly define incompressibility of ensembles of strings S with respect to time complexity classes such as PTIME. In fact, the accepting function for membership in a "hard" set in a time-complexity class $\mathfrak{I}$ behaves "randomly" when viewed as input to a Turing machine that has a more restricted amount of time available. This idea goes back at least to Meyer and McCreight (1971).

The three principles all involve some notion of *functional composition* or of iteration of a function. Indeed, running a computation of a Turing machine can be viewed as a kind of functional composition with feedback. The first two principles directly lead to candidates for number-theoretic pseudorandom number generators (see below). The computational complexity principle leads to pseudorandom number generators having provably good properties with respect to some level of computational resources, but most of these are not of use in practice because one must use more than the allowed bound on computation to find them in the first place. Finally, we note that there are very close relations between expansiveness properties of mappings and the Kolmogorov-complexity of descriptions of their trajectories (Brudno, 1982).

A variety of functions have been proposed for use in pseudorandom bit generators. We give several examples below. Most of these functions are number-theoretic, and nearly all of them have an underlying group structure.

EXAMPLE 1 (Multiplicative congruential generator). The generator is defined by

$$x_{n+1} \equiv ax_n + b \pmod M,$$

where $0 \leq x_n \leq M - 1$. Here (a, b, M) are the parameters describing the generator, and x_0 is the seed. This was one of the first proposed pseudorandom number generators. Generators of this form are widely used in practice in Monte Carlo methods, taking x_i/M to simulte uniform draws on $[0, 1]$ (Knuth, 1981; Marsaglia, 1968; Rubinstein, 1982). More generally, one can consider polynomial recurrences (mod M).

EXAMPLE 2 (Power generator). The generator is defined by

$$x_{n+1} \equiv (x_n)^d \pmod N.$$

Here (d, N) are parameters describing the generator, and x_0 is the seed.

An important special case of the power generator occurs when $N = p_1 p_2$ is a product of two distinct odd primes. The first case occurs when $(d, \varphi(N)) = 1$, where

$$\varphi(N) = \#(\mathbf{Z}/N\mathbf{Z})^* = (p_1 - 1)(p_2 - 1)$$

is Euler's totient function that counts the number of

multiplicatively invertible elements (mod N). Then the map $x \to x^d \pmod N$ is one-to-one on $(\mathbf{Z}/N\mathbf{Z})^*$, and this operation is the encryption operation of the RSA public-key cryptosystem (Rivest, Shamir and Adleman, 1978), where (d, N) are publicly known. We call this special case an *RSA generator.*

EXAMPLE 3 (Discrete exponential generator). The generator is defined by

$$x_{n+1} \equiv g^{x_n} \pmod N.$$

Here (g, N) are parameters describing the generator, and x_0 is the seed.

A special case of importance occurs when N is an odd prime p and g is a primitive root (mod p). Then the problem of recovering x_n given (x_{n+1}, g, p) is the *discrete logarithm problem*, an apparently hard number-theoretic problem (Odlyzko, 1985). The discrete exponentiation operation (mod p) was suggested for cryptographic use in the key exchange scheme of Diffie and Hellman (1976). A *key exchange scheme* is a method for two parties to agree on a secret key using an insecure channel. Blum and Micali (1984) gave a method for generating a set of pseudorandom bits whose security rests on the difficulty of solving the discrete logarithm problem.

EXAMPLE 4 (Kneading map). Consider a bivariate transformation

$$(x_{n+1}, y_{n+1}) := (y_n, x_n + f(y_n, z_n)),$$

where f is a fixed bivariate function, usually taken to be nonlinear. The function $f(\cdot,\cdot)$ determines the generator, and (x_0, y_0) and the family $\{z_n\}$ constitute the seed. One often takes $z_n := K$ for all integers n, for a fixed K. This mapping has the feature that it is one-to-one, with inverse

$$(x_n, y_n) := (y_{n+1} - f(x_{n+1}, z_n), x_{n+1}).$$

One can generalize this construction to take x, y and f$(\cdot,\cdot)$ to be vector-valued. The *data encryption standard* (*DES*) cryptosystem is composed of 16 iterations of (vector-valued) maps of this type, where (x_0, y_0) are the data to be encrypted, all $z_i = K$ compose the key, and f is a specific nonlinear function representable as a polynomial in several variables.

Other examples include the 1/P-generator and the square generator studied in Blum, Blum and Shub (1986) and cellular automata generators proposed by Wolfram (1986).

In addition to these examples, more complicated pseudorandom number generators can be built out of them using the following two mixing constructions.

CONSTRUCTION 1 (Hashing). If $\{x_n\}$ are binary strings of k bits and $H: \{0,1\}^k \to \{0,1\}^l$ is a fixed function, called in this context a *hash function*, then define $\{z_n\}$ by

$$z_n = H(x_n).$$

The traditional use of hash functions is for data compression, in which case l is taken much smaller than k. Sets of good hash functions play an important role in the theory of pseudorandom generators, where they are used to convert highly nonuniform probability distributions on k inputs into nearly uniform distributions on l inputs; see, for instance, Goldreich, Krawczyk and Luby (1988).

A very special case of this construction is the *truncation operator*

$$T_{j,l}(x) = [2^{-j}x] \pmod{2^l},$$

which extracts from a k-bit string x the substring of l bits starting j bits from its right end. In its most extreme form, $T(x) \equiv x \pmod 2$ extracts the least significant bit. The truncation operation was originally suggested by Knuth to be applied to linear congruential generators to cut off half their bits (saving the high-order bits) as a way of increasing apparent randomness. This idea was used in a number of schemes for encrypting passwords to computer systems. However, it is insecure. Reeds (1979) successfully cryptanalyzed one such system.

CONSTRUCTION 2 (Composition). Let $\{x_n\}$ and $\{y_n\}$ be sequences of k-bit strings, let $* : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ be a binary operation, and set

$$z_n = x_n * y_n.$$

The simplest case occurs with $k = 1$, where $*$ is the XOR operation

$$z_n := x_n + y_n \pmod 2,$$

where z_n, x_n, and y_n are viewed as k-dimensional vectors over $GF(2)$. More generally, whenever the operation table of $*$ on $\{0,1\}^k \times \{0,1\}^k$ forms a Latin square (but is not necessarily either commutative or associative), then the $*$ operation has certain randomness-increasing properties when the elements $\{x_n\}$ and $\{y_n\}$ are independent draws from fixed distributions Q and $\bar{Q}$ on $\{0,1\}^k$ (Marsaglia, 1985).

There is a constant search for new and better practical pseudorandom number generators. Recently, Marsaglia and Zaman (1991) proposed some simple generators that have extremely long periods; whether they have good statistical properties remains to be determined.

3. COMPUTATIONAL INFORMATION THEORY AND CRYPTOGRAPHY

In the last few years, there has been extensive development of a theoretical basis of secure pseudorandom number generators. This area is called *computational information theory.*

Computational information theory represents a blending of information theory and computational complexity theory (Yao, 1988). The basic notion taken from information theory is *entropy*, a measure of information. Here, "information" equals "randomness." Information theory was developed by Shannon in the 1940s, and a relationship between computational complexity and information theory was first observed by Kolmogorov. Both Shannon and Kolmogorov assumed in defining "amount of information" that unbounded computational resources were available to extract that information from the data. In contrast, computational information theory assumes that the amount of computational resources is bounded by a polynomial in the size of the input data.

Here we indicate the ideas behind the theory, assuming some familiarity with the basic concepts of computational complexity theory for uniform models of computation (Turing machines) and for nonuniform models of computation (Boolean circuits). Turing machines and polynomial-time computability are discussed in Garey and Johnson (1979), and circuit complexity is described in Savage (1976) and Karp and Lipton (1982).

What is a pseudorandom number generator? The basic idea of pseudorandomness is to take a small amount of randomness described by a seed drawn from a given source probability distribution and to produce deterministically from the seed a larger number of apparently random bits that simulate a sample drawn from another target probability distribution on a range space. That is, it is *randomness-increasing.*

The amount of randomness in a probability distribution is measured by its *binary entropy* (or *information*), which for a discrete probability distribution P is

$$H(P) = -\sum_x p(x) \log_2 p(x),$$

where x runs over the atoms of P. In particular,

$$H(U_k) = k;$$

that is, "k coin flips give k bits of information." The notion of randomness-increasing is impossible in classical information theory because *any deterministic mapping G applied to a discrete probability distribution P never increases entropy*; that is,

$$H(G(P)) \le H(P).$$

However, this may be possible when computing power is limited. Indeed, what may happen is that $G(P)$ may approximate a target distribution Q having a much higher entropy so well that, within the limits of computing power available, one cannot tell the distributions $G(P)$ and Q apart. If $H(Q)$ is much larger than $H(P)$, then we can say that G is *computationally randomness-increasing.*

The fundamental principle is that a *definition of pseudorandom numbers is always relative to the use to which the pseudorandom numbers are to be put.* This use is to simulate the target probability distribution to within a specified degree of approximation.

We measure the degree of approximation using statistical tests. Let P be the source probability distribution, $G(P)$ the probability distribution generated by the pseudorandom number generator G and Q the target probability distribution to be approximated by $G(P)$. A *statistic* is any deterministic function $\sigma(x)$ of a sample x drawn from a distribution, and a *statistical test* σ consists of computation of a statistic σ drawn from $G(P)$. We say that distributions P_1 and P_2 on the same sample space $\mathcal{S}$ are *ε-indistinguishable using the statistic σ* on $\mathcal{S}$ provided that the expected values of $\sigma(x)$ drawn from P_1 and P_2, respectively, agree to within the tolerance level ε; that is,

$$|E[\sigma(x) : x \in P_1] - E[\sigma(x) : x \in P_2]| \le \varepsilon.$$

Given a collection $\mathfrak{I} = \{(\sigma_i, \varepsilon_i)\}$ of statistical tests σ_i with corresponding tolerance levels ε_i, we say that G is a *$\mathfrak{I}$-pseudorandom number generator* from source P to target Q provided that $G(P)$ is ε_i-indistinguishable from Q for all the statistical tests σ_i drawn from $\mathfrak{I}$.

To make these ideas clearer, we allow as source distribution the uniform distribution U_k on the set $\{0, 1\}^k$ of binary strings of length k, and for target distributions we allow either U_ℓ for some ℓ, or else $U([0, 1]^\ell)$, which is the distribution of ℓ independent draws from the uniform distribution on $[0, 1]$.

As an example, consider the linear congruential generator $x_{n+1} \equiv ax_n + b \pmod{M}$ with $M = 2^k - 1$, where the seed x_0 is drawn from $0 \le x_0 \le 2^k - 1$ and is viewed as an element of $\{0,1\}^k$ drawn with distribution U_k. We use this generator to produce a vector of ℓ iterates, $G(x_0) = (x_1/M, x_2/M, \ldots, x_\ell/M)$ and view $G(U_k)$ as approximating the distribution $U([0, 1]^\ell)$. Various statistics that one might consider for $y = (y_1, \ldots, y_\ell)$ in the sample space $\mathcal{S} = [0, 1]^\ell$ are

1. *Average*: $\sigma_A(y) = \frac{1}{\ell} \sum_{i=1}^{\ell} y_i$
2. *Maximum*: $\sigma_M(y) = \max\{y_i : 1 \le i \le \ell\}$
3. *Maximum run*: $\sigma_R(y) = \max\{j : y_i < y_{i+1} < \cdots < y_{i+j} \text{ for some } i\}$
4. *mth autocorrelation*: $\sigma_m(y) = \frac{1}{\ell - m} \sum_{i=1}^{\ell-m} y_i y_{i+m}.$

It is now a purely mathematical problem to decide what ε-tolerance level can be achieved for each of these statistics, viewing $G(U_k)$ as approximating $U([0, 1]^\ell)$. Various results for such statistics for linear congruential generators can be found in Niederreiter (1978).

The statistical tests applied to pseudorandom num-

ber generators for use in Monte Carlo simulations are generally simple; for cryptographic applications, psuedorandom number generators must pass a far more stringent battery of statistical tests.

Now we are ready to give a precise definition of a secure pseudorandom bit generator. To achieve insensitivity to the computational model, this definition, due to Yao (1982), is an asymptotic one. Instead of a single generator G of fixed size, we consider an ensemble $\mathcal{G} = \{G_k : k \geq 1\}$ of generators with

$$G_k : \{0,1\}^k \to \{0,1\}^{f(k)}$$

that have the property of being *polynomial length-increasing*; that is

$$k + 1 \leq f(k) \leq k^m + m$$

for some fixed integer $m \geq 1$. We say that $\mathcal{G}$ is a *nonuniform secure pseudorandom bit generator* (*N-PRG*) if:

1. there is a (deterministic) polynomial-time algorithm that computes *all* $G_k(\mathbf{x})$, given $(k, \mathbf{x})$ as input with $\mathbf{x} \in \{0,1\}^k$, and
2. for any PSIZE family $\mathcal{F}$ of Boolean circuit statistical tests, $G_k(U_k)$ is *polynomially indistinguishable* by $\mathcal{F}$ from $U_{f(k)}$; that is, for all $m \geq 1$, the distribution $G_k(U_k)$ is k^{-m}-indistinguishable by $\mathcal{F}$ from $U_{f(k)}$ for all sufficiently large k.

Condition 1 says that $\mathcal{G}$ is easy to compute. Condition 2 requires more explanation. A *Boolean circuit* C_ℓ is made of *and*, *or* and *not* gates having ℓ inputs and computes a Boolean function $\hat{C}_\ell : \{0,1\}^\ell \to \{0,1\}$. A family $\mathcal{F} = \{C_k : k \geq 1\}$ of Boolean circuits is of *polynomial size* (abbreviated $\mathcal{F} \in$ PSIZE) if there is some fixed m such that the number of gates in C_k is $\leq k^m + m$, for all m. The Boolean function $C_\ell(x)$ computed by C_ℓ with $\ell = f(k)$ provides a statistical test for comparing $G_k(U_k)$ and U_ℓ. Then Condition 2 asserts that

$$\left| E[C_k(x) : x \in G_k(U_k)] - E[C_k(x) : x \in U_{f(k)}] \right| < k^{-m},$$

$$\text{for all } k > c_0(\mathcal{F}, m).$$

The PSIZE measure of computational complexity is *nonuniform* in that no relation between the circuits C_k for different k is required. In particular, the family $\mathcal{F}$ might be noncomputable (nonrecursive). Condition 2 may be colloquially rephrased as, "$\mathcal{G}$ passes all polynomial size statistical tests."

This definition implies that N-PRGs must have a *one-way property*: Given the output $\mathbf{y} = G_k(\mathbf{x}) \in \{0,1\}^{f(k)}$, one cannot recover the string $\mathbf{x} \in \{0,1\}^k$ using a PSIZE family of circuits for a fraction of the inputs exceeding $1/(k^m + m)$ for any fixed m. Otherwise, it would fail the statistical test, "Is there a seed $\mathbf{x}$ such that $G_k(\mathbf{x}) = \mathbf{y}$?"

We also consider the weaker notion of *uniform secure pseudorandom bit generator* (U-PRG) in which PSIZE in Condition 2 is replaced by PTIME, where a family $\mathcal{F} = \{C_k\}$ of circuits is in PTIME if there is a polynomial-time algorithm for computing C_k, given 1^k as input. This condition may be rephrased as, "$\mathcal{G}$ passes all polynomial time statistical tests." Since PTIME $\subseteq$ PSIZE, an N-PRG is always a U-PRG.

These definitions of N-PRG and U-PRG form the basis of a nice theory, whose purpose is to reduce the existence question for apparently very complicated objects (N-PRGs) to the existence question for simpler, more plausible objects. As an example, we have:

THEOREM 1. *If there exists an N-PRG with* $f(x) = k + 1$, *then there exists an N-PRG with* $f(k) = k^m + m$ *for each* $m \geq 2$. *(Similarly for U-PRGs.)*

That is, if there is an PRG that can inflate k random bits to $k + 1$ pseudorandom bits, this is already sufficient to construct a PRG that inflates k random bits to any polynomial number of pseudorandom bits. A proof appears in Boppana and Hirschfeld (1989).

One disadvantage of these definitions (N-PRG, etc.) at present is that *no such generators have been proved to exist*. In fact, proving that they exist is at least as hard a problem as settling the famous $P \neq NP$ question of theoretical computer science. On the positive side, however, Theorem 4 below suggests that the RSA generator may well be a U-PRG.

Yao (1982) provided the basic insight that the nature of secure pseudorandomness (N-PRG) is the *computational unpredictability* of successive bits produced by a pseudorandom bit generator. The notion of computational unpredictability is captured by the concept of a *nonuniform next-bit test*, as follows. Let $\mathcal{G} = \{G_k\}$ be a collection of mappings $G_k : \{0,1\}^k \to \{0,1\}^{f(k)}$, where $f(k)$ is polynomial in k. A *predicting collection* is a doubly indexed family of Boolean circuits $\mathcal{C} = \{C_{i,k} : 1 \leq i \leq f(k) - 1, k \geq 1\}$ computing functions $\hat{C}_{i,k} : \{0,1\}^i \to \{0,1\}$, where the circuit $C_{i,k}$ is viewed as predicting the $(i+1)$st bit of a sequence in $\{0,1\}^{f(k)}$ given the first i bits. The predicting collection is *polynomial size* if there is some $m \geq 2$ such that each $C_{i,k}$ has at most $k^m + m$ gates. Let $p^{\mathcal{C}}_{i,k}$ denote the probability that the output of $\hat{C}_{i,k}$ applied to a bit sequence $(x_1, \ldots, x_i)$ drawn from $G_k(U_k)$ is x_{i+1}. We say that $\mathcal{G}$ *passes the nonuniform next-bit test* if for each polynomial-size predicting collection $\mathcal{C}$ and each $m \geq 2$, for all sufficiently large k,

$$\left| p^{\mathcal{C}}_{i,k} - \frac{1}{2} \right| < \frac{1}{k^m + m}$$

holds for $1 \leq i \leq f(k) - 1$. This test says that x_{i+1} cannot be accurately guessed using PSIZE circuit families with $x_1, \ldots, x_i$ as input.

THEOREM 2 (Yao, 1982). *The following are equivalent:*

1. *A collection* $\mathcal{G} = \{G_k\}$ *passes the nonuniform next-bit test.*
2. *A collection* $\mathcal{G} = \{G_k\}$ *passes all polynomial-size statistical tests.*

In this theorem, there is no restriction on the sizes of circuits needed to compute the functions G_k; for example, they could be of exponential size in k. The direction (2) $\Rightarrow$ (1) is fairly easy to prove because the next-bit test is essentially a family of PSIZE statistical tests. The direction (1) $\Rightarrow$ (2) is harder. A proof of this result is given in Boppana and Hirschfeld (1989).

COROLLARY 1. *A polynomial-time computable collection* $\mathcal{G} = \{G_k\}$ *that passes the nonuniform next-bit test is an N-PRG.*

Pseudorandom bit generators must have a very strong one-way property: from the output of such a generator, it must be computationally infeasible to extract any information at all about its input, for essentially all inputs. *One-way functions* are required only to satisfy a weaker one-way property: they are easy to compute but difficult to invert on a nonnegligible fraction of their instances.

A formal definition of one-way function is as follows: a *nonuniform one-way collection* $\{f_k\}$ consists of a family of functions $f_k : \{0,1\}^k \to \{0,1\}^{p(k)}$, where $k \le p(k) \le k^m + m$ for some fixed m, such that:

1. the ensemble $\{f_k\}$ is uniformly polynomial-time computable, and
2. for each PSIZE circuit family $\{C_k\}$ with $C_k : \{0,1\}^{p(k)} \to \{0,1\}^k$, the functions $\{f_k\}$ are hard to invert on a nonnegligible fraction $1/(k^c + c)$ of their inputs, where c is a fixed positive constant depending on $\{C_k\}$. That is, for $\mathbf{x}$ drawn from the uniform distribution on $\{0,1\}^k$,

$$P\left[f_k(\mathbf{x}) = f_k\left(C_k(f_k(\mathbf{x}))\right)\right]$$

is at most $1 - (k^c + c)^{-1}$ for all large enough k (depending on $\{C_k\}$).

The one-way property embodied in this definition is weaker than that required of an N-PRG in several ways: a function f_k can have an extremely nonuniform probability distribution on its range $\{0,1\}^{p(k)}$, it is required to be hard to invert only on a possibly small fraction $1/(k^c + c)$ of its inputs, and even on those inputs where it is hard to invert, some partial information about its inverse may be easy to obtain.

Impagliazzo, Levin and Luby (1989) showed that one can bootstrap a nonuniform one-way collection into an N-PRG.

THEOREM 3. *The following are equivalent:*

1. *There exists a nonuniform polynomial pseudorandom bit generator (N-PRG).*
2. *There exists a nonuniform one-way collection* $\{f_k\}$ *of functions.*

The implication (1) $\Rightarrow$ (2) is easy, because an N-PRG $\mathcal{G} = \{G_k\}$ is a one-way collection. To prove (2) $\Rightarrow$ (1), it suffices by Theorem 1 to enlarge a set of ℓ random bits to obtain $\ell + 1$ pseudorandom bits, for an infinite set of ℓ. We may reduce to the case that the function f_k is length-preserving by padding its input bits if necessary. The first key idea, due to Goldreich and Levin (1989), is that the Boolean inner product

$$B(\mathbf{x},\mathbf{r}) = \sum_{i=1}^{k} x_i r_i \pmod 2$$

is a *hard-core predicate* for all length-preserving one-way families $f : \{0,1\}^k \to \{0,1\}^k$. That is, the probability distribution of the $(2k+1)$-bit strings $(f(\mathbf{x}), \mathbf{r}, B(\mathbf{x},\mathbf{r}))$ for $(\mathbf{x},\mathbf{r})$ drawn uniformly from $\{0,1\}^{2k}$ is polynomially indistinguishable from $(f(\mathbf{x}), \mathbf{r}, \beta)$, where β is a truly random bit drawn independent of $\mathbf{x}$ and $\mathbf{r}$. We are now done if f is a permutation, because then $(f(\mathbf{x}), \mathbf{r}, \beta)$ would be uniformly distributed on $2k+1$ bits. The difficult case occurs when f is not one-to-one, in which case the set $S_{\mathbf{x}} = \{\mathbf{y} : f(\mathbf{y}) = f(\mathbf{x})\}$ may be large. In this case, $f(\mathbf{x})$ contains only about $k - \log_2|S_{\mathbf{x}}|$ bits of information, and so the entropy of $(f(\mathbf{x}), \mathbf{r}, \beta)$ may be smaller than $2k$, and we gain no computational entropy. The key idea of Impagliazzo, Levin and Luby (1989) is to add in $\log_2|S_{\mathbf{x}}|$ extra bits of information from $\mathbf{x}$ by hashing $\mathbf{x}$ with a random hash function $h : \{0,1\}^k \to \{0,1\}^{|S_{\mathbf{x}}|}$. If one can do this, then the distribution of $(f(\mathbf{x}), \mathbf{r}, h, h(\mathbf{x}), B(\mathbf{x},\mathbf{r}))$ will be polynomially indistinguishable from $(f(\mathbf{x}), \mathbf{r}, h, h(\mathbf{x}), \beta)$, and one extra pseudorandom bit will have been created. The nonuniform model of computation is used in determining the number $\log_2|S_{\mathbf{x}}|$ of bits to hash in the hash function for $\mathbf{x}$.

Håstad (1990) proved that a uniform version of Theorem 3 holds; that is, one may replace "nonuniform" by "uniform" in (1) and (2).

Finally, secure PRGs can be used to construct secure private-key cryptosystems and vice versa. A *private-key cryptosystem* uses a (private) key exchanged by some secure means between two users, the possession of which enables them both to encrypt and decrypt messages sent between them. Encrypted messages should be unreadable to anyone else. They should appear random to any unauthorized receiver, and ideally no statistical information at all should be extractable from the encrypted message. This is seldom achieved in actual cryptosystems, and statistical methods are one of the staples of cryptanalysis. Absolute security can always be achieved by using a key of the same length as the totality of encrypted messages to be exchanged (the "*one-time pad*"). How much security is possible when the key is to be shorter than the messages to be encrypted? An analogy between pseudorandom number generation and private-key

cryptosystems is apparent: the key supplies absolutely random bits from which pseudorandom bits (the encrypted message) are created. This analogy can be made precise. For a statement of the result, see Lagarias (1990, Section 6); some related results appear in Impagliazzo and Luby (1989).

Rompel (1990) showed how to construct a secure authentication scheme based on a one-way function.

Finally, we note that probabilistic algorithms play a fundamental role in the theoretical foundations of modern cryptography, as indicated in Goldwasser and Micali (1984).

4. PERFORMANCE OF CERTAIN PSEUDORANDOM BIT GENERATORS

We now turn to results on the performance of various generators; see Knuth (1981) for general background.

Linear congruential pseudorandom number generators have been extensively studied, and many of their statistical properties carefully analyzed. With proper choice of parameters, they can produce sequences $\{x_k\}$ such that $\{x_k (\mathrm{mod}\ M) : 1 \le k \le n\}$ is close to uniformly distributed on $[1, M]$. Marsaglia (1968) was the first to notice that these generators have certain undesirable correlation properties among several consecutive iterates $(x_n, x_{n+1}, \ldots, x_{n+k})$. Extensive analysis is given in Niederreiter (1978). These generators are apparently unsuitable for cryptographic use (Frieze et al., 1988).

The group-theoretic structure of certain other generators can be used to prove "almost-everywhere hardness" results for such generators, including the following one.

RSA Bit Generator. Given $k \ge 2$ and $m \ge 1$, select odd primes p_1 and p_2 uniformly from the range $2^k \le p_i < 2^{k+1}$, and form $N = p_1 p_2$. Select e uniformly from $[1, N]$ subject to $(e, \varphi(N)) = 1$. Set

$$x_{n+1} \equiv (x_n)^e \ (\mathrm{mod}\ N),$$

and let the bit z_{n+1} be given by

$$z_{n+1} \equiv x_{n+1} \ (\mathrm{mod}\ 2).$$

Then $\{z_n : 1 \le n \le k^m + m\}$ are the pseudorandom bits generated from the seed x_0 of length $2k$ bits.

We consider the problem of *predicting* the next bit z_{n+1} of the RSA generator, given $\{z_1, \ldots, z_n\}$. Alexi et al. (1988) showed that getting any information at all about z_{n+1} is as hard as the (apparently difficult) problem of factoring N. Their argument shows how an oracle that guesses the next bit with a probability slightly greater than 1/2 can be used in an explicit fashion to construct an inversion algorithm.

Theorem 4. *Let (e, N) be a fixed pair, where $(e, \varphi(N)) = 1$ and $N = p_1 p_2$ is odd, with the associated power generator*

$$E(y) \equiv y^e := x \ (\mathrm{mod}\ N).$$

Suppose one is given a $0-1$-valued oracle function $O(x)$ for which

$$O(x) \equiv y \ (\mathrm{mod}\ 2)$$

holds for $(\frac{1}{2} + 1/(k^m + m))\,N$ of all inputs $x \in [0, N-1]$, where $k = \log_2 N$. Then there is a probabilistic polynomial time algorithm that makes $O(k^{C_0 m})$ oracle calls, uses $O(k^{C_0 m})$ "random" bits, halts in $O(k^{C_0 m})$ steps and for any x finds y with probability $\ge 1 - 1/N$, where the probability is taken over all sequences of "random" bits.

This algorithm involves several clever ideas. The first of these is due to Ben-Or, Chor and Shamir (1983), using the fact that the encryption function $E(\cdot)$ respects the group law on $(\mathbf{Z}/N\mathbf{Z})^*$; that is,

$$E(ay) = E(a)E(y).$$

They suppose that one is given an oracle O_I that perfectly recognizes membership in an interval $I = [-\delta N/2, \delta N/2]$ for fixed positive $\delta \le 1/2$; that is,

$$O_I(x) = \begin{cases} 1, & \text{if } y \in I, \\ 0, & \text{if } y \notin I. \end{cases}$$

Define $[x]_N$ to be the least nonnegative residue (mod N), and define $abs_N(x)$ to be the distance of the least (positive or negative) residue from 0 so that

$$abs_N(x) = \begin{cases} [x]_N, & 0 \le x < N/2, \\ N - [x]_N, & N/2 < x < N. \end{cases}$$

Let $par_N(x)$ be the parity of $abs_N(x)$. The clever idea is to generate "random" a, b and attempt to compute the *gcd* of $[ay]_N$ and $[by]_N$ using a modified binary *gcd* algorithm due to Brent and Kung (1983) and Purdy (1983). This algorithm computes the greatest common divisor (r, s) by noting that it is equal to 2 $(r/2, s/2)$ if r, s are both even, to $(r/2, s)$ or $(r, s/2)$ if one is even and to $\frac{1}{2}(r+s)$, $\frac{1}{2}(r-s)$ otherwise. It is easy to check that $\max(|r|, |s|)$ is nonincreasing at each iteration and that it decreases by at least a factor of 3/4 every two iterations, so this algorithm halts in at most $2 \log_{4/3} N$ iterations. The Ben-Or, Chor and Shamir algorithm draws pairs (a, b) of "random" numbers uniformly on $[0, N]$. Such a pair will be called "good" (for the unknown y) if $[ay]_N$ and $[by]_N$ both fall in I. For a good pair the oracle O_I can be used to determine correctly the parity of $[ay]_N$ and $[by]_N$, even though y is unknown. We use the fact that if $w \in I$ is even, then $w/2 \in I$, whereas if it is odd, then $w/2 \in [N/2 - |I|/2, N/2 + |I|/2]$. Hence for a good pair, one has $(a/2)y \in I$ if $[ay]_N$ is even and $(a/2)y \notin I$ if $[ay]_N$ is odd. This holds similarly for $[by]_N$. Thus using the group law

$$par_N([ay]_N) = 1 - O_I\left(E\left[\left(\frac{a}{2}\right)y\right]\right)$$

$$= 1 - O_I\left(E\left[\frac{a}{2}\right]x\right)$$

is calculable. Consequently, one can compute one step of the modified binary *gcd* algorithm to $([ay]_N, [by]_N)$ and obtain new values $([a'y]_N, [b'y]_N)$, with (a', b') known. Now (a', b') is still a good pair, so we can continue to follow the modified binary *gcd* algorithm perfectly and eventually determine $[ly]_N = gcd\ ([ay]_N, [by]_N)$, where l is known. We say that the algorithm succeeds if $[ly]_N = 1$, because in that case $y \equiv l^{-1} \pmod N$ is found. We verify success in this case by checking that $E[l^{-1}] = x$. In all other cases, when (a, b) is not good or when it is good and the *gcd* is not 1, we carry out the above procedure for $2 \log_{4/3} N$ iterations and check the final iterate l to see whether $E[l^{-1}] = x$.

This procedure is a probabilistic polynomial-time algorithm. The pair (a, b) is good with probability at least δ^2. Also, because the distributions of $[ay]_N$, $[by]_N$ are uniform on $[-\delta N/2, \delta N/2]$, and because the probability that $gcd(r, s) = 1$ for such draws approaches $\pi^2/6$ as $|I| \to \infty$, it exceeds a positive constant α for all values of $\{|I|\}$. Thus the success probability for any draw (a, b) is $\geq \delta^2\alpha$, and the expected time until the algorithm succeeds is polynomial in $\log N$.

One immediately sees that this algorithm still works if the oracle O_I is not perfect but is accurate with probability of error less than $(4 \log_{4/3} N)^{-1}$.

Subsequent work of several authors showed how an oracle having a $1/2 + 1/(\log N)^k$ advantage in guessing parity of the smallest RSA bit can be used to simulate a much more accurate oracle and then to simulate an oracle such as O_I above with sufficient accuracy to obtain Theorem 4 (Alexi et al., 1988).

Theorem 4 shows that if the factoring problem is difficult, then recovering any information at all about z_{n+1} given $\{z_1, \ldots, z_n\}$ is hard. In particular, the bits $\{z_k\}$ would then be computationally indistinguishable from i.i.d. coin flips.

One can extract several pseudorandom bits from each of the iterates x_n of the RSA generator. The strongest result to date on this problem can be found in Schrift and Shamir (1990).

The use of the group structure on $(\mathbf{Z}/N\mathbf{Z})^*$ to "randomize" is an example of the concept of an *instance-hiding scheme* discussed by Feigenbaum in this issue.

ACKNOWLEDGMENT

This article is an abridged and revised version of Lagarias (1990), published by the American Mathematical Society.

REFERENCES

ALEXI, W., CHOR, B., GOLDREICH, O. and SCHNORR, C. P. (1988). RSA and Rabin functions: Certain parts are as hard as the whole. *SIAM J. Comput.* 17 194-209.

BEN-OR, M., CHOR, B. and SHAMIR, A. (1983). On the cryptographic security of single RSA bits. In *Proceedings of the 15th Annual Symposium on Theory of Computing* 421-430 ACM Press, New York.

BLUM, L., BLUM, M. and SHUB, M. (1986). A simple unpredictable pseudorandom number operator. *SIAM J. Comput.* 15 364-383.

BLUM, M. and MICALI, S. (1984). How to generate cryptographically strong versions of pseudorandom bits. *SIAM J. Comput.* 13 850-864.

BOPPANA, R. and HIRSCHFELD, R. (1989). Pseudorandom generators and complexity classes. In *Randomness and Computation* (S. Micali, ed.) 1-26. JAI Press, Greenwich, CT.

BRENT, R. P. and KUNG, H. T. (1983). Systolic VLSI arrays for linear time *gcd* computations. In *VLSI 83, IFIP* (F. Anceau and D. J. Aas, eds.) 145-154. North-Holland, Amsterdam.

BRUDNO, A. A. (1982). Entropy and the complexity of the trajectories of a dynamical system. *Trans. Moscow Math. Soc.* 44 127-151.

CHAITIN, G. J. (1966). On the length of programs for computing finite binary sequences. *J. Assoc. Comput. Mach.* 13 547-569.

COLLET, P. and ECKMANN, J.-P. (1980). *Iterated Maps on the Interval as Dynamical Systems*. Birkhäuser, Boston.

DEVROYE, L. (1986). *Nonuniform Random Variate Generation*. Springer, New York.

DIFFIE, W. and HELLMAN, M. (1976). New directions in cryptography. *IEEE Trans. Inform. Theory* 22 644-654.

FEIGENBAUM, J. (1993). Probabilistic algorithms for defeating adversaries. *Statist. Sci.* 8 26-30.

FRIEZE, A., HÅSTAD, J., KANNAN, R., LAGARIAS, J. C. and SHAMIR, A. (1988). Reconstructing truncated integer variables satisfying linear congruences. *SIAM J. Comput.* 17 262-280.

GAREY, M. and JOHNSON, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco.

GOLDREICH, O., KRAWCZYK, H. and LUBY, M. (1988). On the existence of pseudorandom generators. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science* 12-24. IEEE Computer Society Press, Los Alamitos, CA.

GOLDREICH, O. and LEVIN, L. (1989). A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual Symposium on Theory of Computing* 25-32. ACM Press, New York.

GOLDWASSER, S. and MICALI, S. (1984). Probabilistic encryption. *J. Comput. System Sci.* 28 270-299.

HÅSTAD, J. (1990). Pseudo-random generators under uniform assumptions. In *Proceedings of the 22nd Annual Symposium on Theory of Computing* 395-404. ACM Press, New York.

IMPAGLIAZZO, R., LEVIN, L. and LUBY, M. (1989). Pseudorandom number generation from one-way functions. In *Proceedings of the 21st Annual Symposium on Theory of Computing* 12-24. ACM Press, New York.

IMPAGLIAZZO, R. and LUBY, M. (1989). One-way functions are essential for complexity-based cryptography. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science* 230-235. IEEE Computer Society Press, Los Alamitos, CA.

KARP, R. and LIPTON, R. (1982). Turing machines that take advice. *Enseign. Math.* 28 191-209.

KNUTH, D. (1981). *The Art of Computer Programming* 2 *Seminumerical Algorithms* 2nd. ed. Addison-Wesley, Reading, MA.

KNUTH, D. and YAO, A. C. (1976). The complexity of nonuniform random number generation. In *Algorithms and Complexity* (J. F. Traub, ed.) 357–428. Academic, New York.

KOLMOGOROV, A. N. (1965). Three approaches to the concept of "the amount of information." *Problems Inform. Transmission* 1 3–13.

LAGARIAS, J. C. (1990). Pseudorandom number generators in number theory and cryptography. In *Cryptology and Computational Number Theory* (C. Pomerance, ed.) 115–143. Amer. Math. Soc., Providence, RI.

MARSAGLIA, G. (1968). Random numbers fall mainly in the planes. *Proc. Nat. Acad. Sci. USA.* 61 25–28.

MARSAGLIA, G. (1985). A current view of random number generators. In *Computer Science and Statistics: Sixteenth Symposium on the Interface* (L. Billard, ed.) 3–11. North-Holland, Amsterdam.

MARSAGLIA, G. and ZAMAN, A. (1991). A new class of random number generators. *Ann. Appl. Probab.* 1 462–480.

MARTIN-LOF, P. (1966). The definition of random sequences. *Inform. and Control* 9 619–682.

MEYER, A.R. and MCCREIGHT, E. M. (1971). Computationally complex and pseudorandom zero-one valued functions. In *Theory of Machines and Computations* (Z. Kohlavi and A. Paz, eds.) 19–42. Academic, New York.

NIEDERREITER, H. (1978). Quasi-Monte Carlo methods and pseudorandom numbers. *Bull. Amer. Math. Soc.* 84 957–1041.

ODLYZKO, A. M. (1985). Discrete logarithms in finite fields and their cryptographic significance. In *Advances in Cryptology: Eurocrypt '84. Lecture Notes in Comput. Sci.* 209 224–316. Springer, New York.

PURDY, G. B. (1983). A carry-free algorithm for finding the greatest common divisor of two integers. *Comput. Math. Appl.* 9 311–316.

REEDS, J. (1979). Cracking a multiplicative congruential encryption algorithm. In *Information Linkage between Applied Mathematics and Industry* (P. C. Wong, ed.) 467–472. Academic, New York.

RIVEST, R., SHAMIR, A. and ADLEMAN, L. (1978). On digital signatures and public key cryptosystems. *Comm. ACM* 21 120–126.

ROMPEL, J. (1990). One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual Symposium on Theory of Computing* 387–394. ACM Press, New York.

RUBINSTEIN, R. Y. (1982). *Simulation and the Monte Carlo Method.* Wiley, New York.

SAVAGE, J. (1976). *The Complexity of Computing.* Wiley, New York.

SCHRIFT, A. and SHAMIR, A. (1990). The discrete log is very discreet. In *Proceedings of the 22nd Annual Symposium on Theory of Computing* 405–415. ACM Press, New York.

SCHUSTER, H. G. (1988). *Deterministic Chaos.* VCH Verlagsgesellschaft GmbH, Weinheim, Germany.

WOLFRAM, S. (1986). Random sequence generation by cellular automata. *Adv. in Appl. Math.* 7 123–169.

YAO, A. (1982). Theory and applications of trapdoor functions (extended abstract). In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science* 80–91. IEEE Computer Society Press, Los Alamitos, CA.

YAO, A. (1988). Computational information theory. In *Complexity in Information Theory* (Y. Abu-Mostafa, ed.) 1–15. Springer, New York.