J. Cryptology (1991) 3: 157–172

Journal of Cryptology

Efficient, Perfect Polynomial Random Number Generators[1]

S. Micali
Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139, U.S.A.

C. P. Schnorr
Fachbereich Mathematik/Informatik, Universität Frankfurt, Robert-Mayer-Strasse 6–10, D-6000 Frankfurt a.M., Germany

Abstract. Let N be a positive integer and let $P \in \mathbb{Z}[x]$ be a polynomial that is nonlinear on the set $\mathbb{Z}_N$ of integers modulo N. If, by choosing x at random in an initial segment of $\mathbb{Z}_N$, $P(x) \pmod N$ appears to be uniformly distributed in $\mathbb{Z}_N$ to any polynomial-time observer, then it is possible to construct very efficient pseudorandom number generators that pass any polynomial-time statistical test. We analyse this generator from two points of view. A complexity theoretic analysis relates the perfectness of the generator to the security of the RSA-scheme. A statistical analysis proves that the least-significant bits of $P(x) \pmod N$ are statistically random.

Key words. Random number generator, Perfect generator, Polynomial generator, RSA-scheme.

1. Introduction

The concept of a *perfect* random number generator (RNG) was introduced by Blum and Micali (1982) and Yao (1982). An RNG is *perfect* if it passes all polynomial-time statistical tests, i.e., the distribution of output sequences cannot be distinguished by a polynomial-time observer 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 RNGs have been established, for example, based on the discrete logarithm by Blum and Micali (1982, 1984), based on quadratic residuosity by Blum *et al.* (1986), based on one-way functions by Yao (1982), and based on RSA-encryption and factoring by Alexi *et al.* (1984, 1988). This last generator is the most efficient of these generators. It successively generates $\log n$ pseudorandom bits by one modular multiplication with a modulus N that is n bit long.

[1] Date received: August 8, 1988. Date revised: October 26, 1990. This research was performed while C. P. Schnorr was visiting the Department of Computer Science of the University of Chicago, who also supported his research. A U.S. patent, based on this work, has been granted.

In this paper we introduce two polynomial RNGs. The first of these generators uses a random RSA-modulus and the second uses an arbitrary prime modulus N. These generators are perfect if the answer to the following question **Q** is "yes" in the corresponding case (of random RSA-moduli (resp. arbitrary prime moduli)).

Q. *Let N be an odd positive integer, let $P \in \mathbb{Z}[x]$ be a polynomial that is nonlinear on $\mathbb{Z}_N$, and let x be chosen at random in an initial segment of $\mathbb{Z}_N$. Does $P(x) \pmod N$ (or a major part thereof) appear to be uniformly distributed to any polynomial-time observer?*

We show that weak variants of this question have a positive answer. We analyse the question from the complexity-theoretic and the statistical point of view. Our complexity-theoretic analysis applies to random RSA-moduli N and to RSA-functions $P(x) = x^e$. It relies on the assumption that the RSA-cryptosystem is secure. We make new use of the analysis of RSA-functions by Alexi *et al.* (1984, 1988). Our statistical analysis applies to arbitrary prime moduli N and arbitrary polynomials P that are nonlinear on $\mathbb{Z}_N$. We prove that the least-significant bits of $P(x) \pmod N$ are statistically random. This follows from an analysis by Niederreiter (1988) of the serial test applied to nonlinear congruential generators.

We believe that our question is important in its own right. As a further incentive to settle it we show that answering it in the affirmative yields *perfect* and *efficient* pseudorandom number generators. We hope that this paper encourages research in this question.

This paper is organized as follows. In Section 2 we recall the fundamental concept of a perfect RNG, we formulate RSA-security and describe previous RNGs that are based on the assumption of RSA-security. In Section 3 we introduce our new sequential polynomial generator with random RSA-moduli. Its security is related to question $\mathbf{Q}_1$, the variant of question **Q** refering to random RSA-moduli. We show that a weak variant of $\mathbf{Q}_1$ can be answered in the affirmative if the RSA-cryptosystem is secure. Assuming that $\mathbf{Q}_1$ has a positive answer, is actually equivalent to assuming that RSA-encryption is secure in a strong sense. In Section 4 we introduce a sequential polynomial generator with prime moduli. Its security is related to question $\mathbf{Q}_2$, a variant of question **Q** adapted to prime moduli. Our statistical analysis shows that a weak variant of $\mathbf{Q}_2$ can actually be answered in the affirmative. In Section 5 we show that if $\mathbf{Q}_1$ has a positive answer, then the sequential polynomial generator is perfect. This generator is more efficient than previous ones based on the assumption that the RSA-scheme is secure, even in a strong sense. We show how to parallelize arbitrary perfect RNGs and how to retrieve substrings of the output efficiently.

2. Perfect RNGs and the RSA-Scheme

We recall the concept of perfect RNG that was introduced by Blum and Micali (1982, 1984) and Yao (1982). We define RSA-security and describe the RNG of Alexi *et al.* (1988) that is perfect if RSA is secure.

An *ensemble* $X = (X_n)_{n \in \mathbb{N}}$ is a sequence of random variables such that, for some function $h: \mathbb{N} \to \mathbb{N}$, X_n ranges over $\{0, 1\}^{h(n)}$. We call h the *type* of the ensemble and we require that h is polynomially bounded, i.e., $h(n) = O(n^t)$ for some $t > 0$. A *poly-size circuit family* $C = (C_n)_{n \in \mathbb{N}}$ is a sequence of boolean circuits C_n such that the number of gates in C_n is at most $O(n^t)$ for some $t > 0$. For a random variable X_n ranging over input values for C_n we let $C_n(X_n)$ denote the induced random variable of output values.

Definition. Two ensembles $(X_n)_{n \in \mathbb{N}}$ and $(Y_n)_{n \in \mathbb{N}}$ are *polynomially indistinguishable* if, for every poly-size circuit family $(C_n)_{n \in \mathbb{N}}$ and for all $t > 0$,

$$|\text{prob}(C_n(X_n) = 1) - \text{prob}(C_n(Y_n) = 1)| = O(n^{-t}).$$

This definition seems to cover all efficient ways to tell two ensembles apart. No additional power can be gained by circuits with many output gates and by circuits that have additional input gates for internal coin flips. The circuit family $(C_n)_{n \in \mathbb{N}}$ is called a *statistical test.*

Let U_n denote the random variable that ranges uniformly on $\{0, 1\}^n$. An ensemble $(X_n)_{n \in \mathbb{N}}$ of type h is called *pseudorandom* if it is polynomially indistinguishable from the ensemble $(U_{h(n)})_{n \in \mathbb{N}}$. More generally we call an ensemble $(X_n)_{n \in \mathbb{N}}$ pseudorandom in the sequence of finite sets $(S_n)_{n \in \mathbb{N}}$ if it is polynomially indistinguishable from the sequence of uniform distributions on $(S_n)_{n \in \mathbb{N}}$. This assumes a natural encoding of the elements in S_n as bit strings of some length $h(n)$.

An RNG is a polynomial-time algorithm $G: \{0, 1\}^* \to \{0, 1\}^*$ such that $G(\{0, 1\}^n) \subset \{0, 1\}^{h(n)}$ for some function h with $h(n) > n$. With an RNG we associate the random variable $G(U_n)$ defined by

$$\text{prob}[G(U_n) = y] = |\{x \in \{0, 1\}^n : G(x) = y\}| 2^{-n}.$$

Definition. An RNG G is called *perfect* if the ensemble $(G(U_n))_{n \in \mathbb{N}}$ is pseudorandom.

An ensemble $(X_n)_{n \in \mathbb{N}}$ is called *unpredictable* (to the right) if, for every poly-size circuit family C and for all $t > 0$,

$$\max_{i=0,\ldots,h(n)-1} |\text{prob}[C_{n,i}(x_1 \cdots x_i) = x_{i+1}] - \tfrac{1}{2}| = O(n^{-t}),$$

where $x_1 \cdots x_{h(n)} \in_{X_n} \{0, 1\}^{h(n)}$, i.e., the random variable $x_1 \cdots x_{h(n)}$ has the same distribution as X_n.

Theorem 2.1 (Yao, 1982). *An ensemble is pseudorandom iff it is unpredictable.*

Alexi *et al.* have shown that unpredictable RNGs can be derived from the RSA-algorithm provided that the RSA-scheme is secure. We describe this RNG. The RSA-scheme of Rivest *et al.* is based on the subgroup $\mathbb{Z}_N^*$ of invertible elements in $\mathbb{Z}_N = \mathbb{Z}/N\mathbb{Z}$ where N is a product of two large random primes p and q. We identify $\mathbb{Z}_N$ with the integer interval $[1, N]$. For $z \in \mathbb{Z}$ let $[z]_N \in [1, N]$ denote the least

positive residue of z modulo N; we have $[z]_N = z \pmod N$. The order of the group $\mathbb{Z}_N^*$ is $\varphi(N) = (p-1)(q-1)$. If e is an integer, $e > 1$, and $\gcd(e, \varphi(N)) = 1$, then the transformation

$$[1, N] \ni x \mapsto [x^e]_N \in [1, N] \tag{1}$$

is a permutation on $[1, N]$ and is called an RSA-cipher. The inverse permutation to (1) is given by $x \mapsto [x^d]_N$ where $d = e^{-1} \pmod{\varphi(N)}$.

The security of the RSA-scheme relies on the assumption that the RSA-cipher is difficult to invert when e, N are given but $\varphi(N)$ and $d = e^{-1} \pmod{\varphi(N)}$ are unknown. All known methods for inverting the RSA-cipher require the factorization of N. We formulate RSA-security with respect to uniform random moduli N from the set

$$S_{n,e} = \left\{ N \in \mathbb{N} \;\middle|\; \begin{array}{c} N = p \cdot q,\ \gcd(e, \varphi(N)) = 1 \\ 2^{n/2-1} < p, q < 2^{n/2} \end{array} \right\}.$$

The set $S_{n,e}$ is sufficiently large. It follows from the prime number theorem and from Dirichlet's theorem on the distributions of primes in arithmetic progressions that

$$\lim_n |S_{n,e}| 2^{-n} n^2$$

is a positive constant.

We call the RSA-scheme *secure* if, for every poly-size circuit family C and all $t > 0$,

$$\operatorname{prob}(C_n([x^e]_N, N) = x) = O(n^{-t}).$$

The probability space is the set of all pairs (x, N) with $N \in S_{n,e}$ and $x \in [1, N]$ with uniform probability.

Notation. For $x, k \in \mathbb{N}$ we let $x[k$ denote the bit string consisting of the k least-significant bits of x. We let $y \in_R S$ denote a random variable y that ranges uniformly on the set S. We let $\log n$ denote the nature logarithm of n.

Theorem 2.2 (Alexi *et al.*, 1984, 1988). *If the RSA-scheme is secure and* $k(n) = O(\log n)$, *then the following ensembles are polynomially indistinguishable*:

- $(N, [x^e]_N, x[k(n))$ *for* $N \in_R S_{n,e}$, $x \in_R [1, N]$.
- (N, y, z) *for* $N \in_R S_{n,e}$, $y \in_R [1, N]$, $z \in_R \{0, 1\}^{k(n)}$.

The "Incestuous" Generator. Theorem 2.2 yields a perfect RNG under the assumption that the RSA-scheme is secure. This generator consists of two phases. In the first phase, a pseudorandom RSA-modulus N and a random x in $[1, N]$ are constructed from a truly random seed ω. In the second phase the pseudorandom bit string is generated by computing the sequence $f(x), f(f(x)), \ldots$ (where $f(x) = [x^e]_N$) and outputing at each step the $k(n)$ least-significant bits of the current value. Thus, at each step, the function f is evaluated on a value comprising the previous output bits, in a somewhat "incestuous" manner.

Phase 1. Given a random bit string $\omega \in_R \{0, 1\}^{n^3}$ generate, from ω by a deterministic polynomial-time algorithm H, a pair $(x, N) = H(\omega)$ consisting of two n bit numbers

x, N such that the following ensembles are polynomially indistinguishable:

- $(H(U_n^3))_{n \in \mathbb{N}}$,
- (y, N) with $N \in_R S_{n,e}$, $y \in_R [1, N]$.

Phase 2. $x_1 := x$, for $i = 1, \ldots, h(n)$ do

$$x_{i+1} := [x_i^e]_N, \textit{ output } \mathrm{Out}(x_i) := x_i[k(n)).$$

Sketch of the Function H. We can generate, from ω random numbers, $p_1, \ldots, p_{n^2}$ in the interval $[2^{n/2-1}, 2^{n/2}]$ such that with overwhelming probability there are at least two primes p_i, p_j satisfying $\gcd((p_i - 1)(p_j - 1), e) = 1$. We can also generate from ω the random bits for a probabilistic primality proof for p_i and p_j and a probabilistic proof showing that $\mathrm{prob}_\omega[N \in S_{n,e}] \geq 1 - 2^{-n}$ holds for $N = p_i p_j$. For this we can use the probabilistic primality test of Solovay and Strassen (1977, 1978). In case there are no such primes p_i, p_j we define $H(\omega)$ arbitrarily. The random number $x \in [1, N]$ can be generated from unused bits of ω.

Corollary 2.3 (Alexi *et al.*, 1984, 1988). *If the RSA-scheme is secure, $k(n) = O(\log n)$ and $h(n)$ is polynomial with $h(n) > n^3$, then the following ACGS-generator is a perfect RNG:*

$$G(\omega) := x_1[k(n)), \ldots, x_{h(n)}[k(n)) \qquad \textit{for} \quad \omega \in \{0, 1\}^{n^3},$$

where $(x, N) = H(\omega)$ is as above and $x_0 = x$, $x_{i+1} = [x_i^e]_N$.

The above generator raises a natural question:

How many bits can we safely output at each iteration of the above generator?

Assuming that $n/2$ bits can be output and the generator is perfect, to have an *expansion factor* of 2 (i.e., to output as many pseudorandom bits as in x and to obtain a new seed x_i) the RSA-function needs to be iterated at least *twice.*

By contrast, in the next section we show a simple, different strategy that obtains an expansion factor 2 with a single function evaluation using half-size input numbers. This generator is of the *weaning* type. Once a string of bits is output it is left alone, that is, the function is never evaluated on the output bits.

3. The Sequential Polynomial Generator of the Weaning Type

The *sequential polynomial generator* (SPG) consists of two phases similar to the phases of the "incestuous" generator. The first phase generates a pseudorandom RSA-modulus N and a random $x \in [1, N2^{-k}]$ from a truly random seed ω. The second phase computes a sequence of numbers $x = x_1, x_2, \ldots, x_i, \ldots$ in $[1, N2^{-k}]$ and outputs from x_i the bit string $\mathrm{Out}(x_i) \in \{0, 1\}^{k(n)}$ consisting of the $k(n)$ least-significant bits of $z_i := [x_i^e]_N$. The *successor* x_{i+1} of x_i is formed from a *separate* part of z_i, the remaining $n - k(n)$ most-significant bits of z_i.

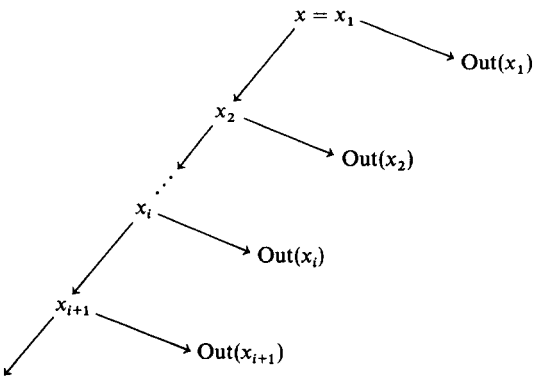

Fig. 1. The sequential polynomial generator (SPG).

Phase 2. $x_1 := x$, for $i = 1, \ldots, h(n)$ do

$$z_i := [x_i^e]_N,\ x_{i+1} := \lfloor z_i/2^{k(n)} \rfloor + 1 \ \textit{Output}\ \text{Out}(x_i) := z_i[k(n)).$$

The SPG can be illustrated by the infinite tree shown in Fig. 1.

Let $\text{SPG}_{h,e}(x, N) = \prod_{i=1}^{h} \text{Out}(x_i)$ (where $\prod$ denotes concatenation) be the output of the first h steps. If $k(n) = n/2$, then a single iteration of this generator yields an *expansion factor* 2, i.e., the output string $\text{Out}(x_1)$ has as many bits as are in x. Assuming that the SPG is perfect with $k(n)$ is comparable to the assumption that the "incestuous" generator is perfect with $k(n)$. In particular we show, in Corollary 5.2, that the SPG is perfect with $k(n) = O(\log n)$ if the RSA-scheme is secure.

We study the perfectness of the SPG in view of the following complexity question $\mathbf{Q}_1$, the variant of question $\mathbf{Q}$ adapted to RSA-moduli. We prove in Section 5 that if $\mathbf{Q}_1$ has a positive answer for $k(n)$, then the above generator is perfect.

$\mathbf{Q}_1$. *Let* $k: \mathbb{N} \to \mathbb{N}$ *be a function where* $k = k_e$ *can depend on* e. *Is it true that the following ensembles are polynomially indistinguishable?*

- $(N, [x^e]_N)$ *for* $N \in_R S_{n,e}$ *and* $x \in_R [1, N2^{-k(n)}]$.
- (N, y) *for* $N \in_R S_{n,e}$ *and* $y \in_R [1, N]$.

The question may be interesting in its own right, since it addresses a number theoretic problem that reaches far beyond RSA-ciphers. In the next section we study a corresponding question for prime moduli N and for arbitrary nonlinear polynomials. In this section we study $\mathbf{Q}_1$ from the point of view of RSA-security.

If the answer to $\mathbf{Q}_1$ is no for $k(n)$, then given e and N we can distinguish between RSA-ciphertexts $[x^e]_N$ of random messages $x \in_R [1, N]$ and RSA-ciphertexts of messages $x \in_R [1, N2^{-k(n)}]$. Then RSA-ciphertexts leak partial information. This does not necessarily jeopardize the RSA-scheme since short messages $x \in [1, N2^{-k}]$ can be avoided.

The following theorem relates $\mathbf{Q}$ and $\mathbf{Q}_1$ more closely to RSA-security. We say that a probabilistic algorithm A ε_N*-rejects* the distribution D on $[1, N]$ if

$$|p_A - \bar{p}_A| \geq \varepsilon_N,$$

where p_A (resp. $\bar{p}_A$) is the probability that A on input $y \in [1, N]$ outputs 1. The probability space is the set of all $y \in [1, N]$, distributed according to D (resp. with uniform distribution) and of all 0–1 sequences of internal coin tosses of algorithm A.

Theorem 3.1. *Let $N \in S_{n,e}$. Every probabilistic algorithm A that ε_N-rejects RSA-ciphertexts $[x^e]_N$ of random messages $x \in [1, N2^{-k}]$, can be transformed (uniformly in N) into a probabilistic algorithm for decoding arbitrary RSA-ciphertexts. This deciphering algorithm terminates after at most $(2^k \varepsilon_N^{-1} n)^{O(1)}$ elementary steps and succeeds with probability at least 1/2.*

For *elementary* steps we count $\mathbb{Z}_N$-operations (addition, multiplication, division), RSA-encryptions, and calls to the algorithm A at unit costs.

Proof. For odd N and all $x \in [1, N]$ we have

$$x \in [1, N2^{-k}] \quad \Leftrightarrow \quad [2^k x]_N = 0 \pmod{2^k}$$

(i.e., $x \in [1, N2^{-k}]$ iff the representative of $2^k x \pmod N$ in $[1, N]$ is a multiple of 2^k). Therefore the following two distributions are identical for odd N, put $y = [2^k x]_N$:

- $[x^e]_N$ for random $x \in [1, N2^{-k}]$,
- $[2^{-ke} y^e]_N$ for random $y \in [1, N]$ satisfying $y = 0 \pmod{2^k}$.

Moreover we can transform in polynomial time $y^e \pmod N$ into $2^{-ke} y^e \pmod N$. Thus an ε_N-rejection of RSA-ciphertexts $[x^e]_N$ of random messages $x \in [1, N2^{-k}]$ can be transformed (uniformly in N) into an ε_N-rejection of RSA-ciphertexts $[y^e]_N$ of random messages y satisfying $y = 0 \pmod{2^k}$.

For random $y \in [1, N]$ the event $y = 0 \pmod{2^k}$ has probability at least $2^{-k} - 1/N$. Therefore an ε_N-rejection of RSA-ciphertexts $[y^e]_N$ of random messages y satisfying $y = 0 \pmod{2^k}$ translates into an $\varepsilon_N 2^{-k}$-rejection of pairs $([y^e]_N, y \pmod{2^k})$ of random messages $y \in_R [1, N]$. By Theorem 2.2 $y \pmod{2^k}$ is pseudorandom when given $[y^e]_N$, or else RSA is insecure. Following the proof (in Alexi *et al.*, 1984, 1988) of Theorem 2.2 we can transform an $\varepsilon_N 2^{-k}$-rejection of the pairs $([y^e]_N, y \pmod{2^k})$ into a probabilistic algorithm for inverting RSA which terminates after at most $(2^k \varepsilon_N^{-1} n)^{O(1)}$ elementary steps and succeeds with probability at least 1/2. □

We state consequences of Theorem 3.1 to the question $\mathbf{Q}_1$ for various functions $k(n)$.

The Case $k(n) = O(\log n)$. If $k(n) = O(\log n)$, then the time bound $(2^k \varepsilon_N^{-1} n)^{O(1)}$ of the RSA-deciphering algorithm in Theorem 3.1 is polynomial in the time bound of algorithm A. Therefore if the answer to $\mathbf{Q}_1$ is no for $k(n) = O(\log n)$, then RSA-ciphertexts can be deciphered in probabilistic polynomial time.

The Case $k(n) = O(n^{1/3})$. If $\mathbf{Q}_1$ is answered in the negative for $k(n) = O(n^{1/3})$, then RSA-ciphertexts can be deciphered with probability at least 1/2 in time $e^{O(n^{1/3})}$. This is faster than the fastest known algorithm for RSA-deciphering via factoring. (So far the fastest algorithm for factoring RSA-moduli requires $e^{O(n \log n)^{1/2}}$ steps. The recently invented number field sieve algorithm by Lenstra *et al.* (1990) takes

$e^{O(n \log^2 n)^{1/3}}$ steps to factor numbers of the form $r^e \pm s$, where s is a small integer.) Thus if $\mathbf{Q}_1$ is answered in the negative for $k(n) = O(n^{1/3})$, then we can speed up the presently known attacks to the RSA-scheme.

The Case $k(n) = n(1 - 1/e)$ and $k(n) = n(1 - 2/e)$. The answer to $\mathbf{Q}_1$ for $k(n) = n(1 - 1/e)$ and $N2^{-k(n)} = N^{1/e}$ is "no" since we have $x^e \leq N$ for all $x \in [1, N^{1/e}]$ and thus RSA-ciphertexts $x^e \pmod N$ can be easily deciphered. On the other hand, the question $\mathbf{Q}_1$ remains open for $k(n) = n(1 - 2/e)$, $N2^{-k(n)} = N^{2/e}$. The numbers x^e are of order N^2 for almost all $x \in [1, N^{2/e}]$. This may be sufficient to make the task of deciphering $x^e \pmod N$ hard. At least it is known that inverting the squaring $x \mapsto x^2 \pmod N$ is as hard as factoring N, and the squares x^2 are of order N^2, too. If the answer to $\mathbf{Q}_1$ is yes for $k(n) = n(1 - 2/e)$, then this yields a particular efficient perfect RNG, see Section 5.

Choosing the Modulus Size. In order that the two distributions in question $\mathbf{Q}_1$ are infeasible to distinguish, N must be difficult to factor and $N2^{-k}$ must be large enough so that it is infeasible to enumerate the interval $[1, N2^{-k}]$. We must also take into account the following method by Pollard (1988) to recover $x \in [1, N2^{-k}]$ from $x^e \pmod N$. Let $N2^{-k} = N^{2\alpha}$. If x is a product $x = uv$ of two numbers, $u, v \in [1, N^{\alpha}]$, then we can find u, v as follows:

1. Generate the set $S_1 = \{u^e \pmod N | u \in [1, N^{\alpha}]\}$ and sort this set.
2. Generate the set $S_2 = \{x^e v^{-e} \pmod N | v \in [1, N^{\alpha}]\}$ and sort this set.
3. Test whether S_1 and S_2 have a common element. If $u^e = x^e v^{-e} \pmod N \in S_1 \cap S_2$, then we have found $x = uv$.

Pollard's attack performs $O(N^{\alpha})$ arithmetical steps modulo N and stores N^{α} residues modulo N. If we choose n, k such that $n - k(n) \geq 128$, then Pollard's attack takes about 2^{64} steps which seems to be infeasible for present computers.

4. Statistical Analysis of Generators Using a Prime Modulus

In Theorem 4.1 we analyse a variant of question $\mathbf{Q}_1$ that relates to an arbitrary prime modulus N from the statistical point of view. (This analysis can be extended from prime moduli N to RSA-moduli N, see the remark after the proof of Theorem 4.1.) We also introduce an RNG using an arbitrary prime modulus N. Let N be an arbitrary odd prime. Let $P \in \mathbb{Z}[x]$ be an arbitrary polynomial that is nonlinear when considered modulo N.

Niederreiter (1988) has proved an upper bound (see Lemma 4.2 below) on the discrepancy of the distribution of $P(x) \pmod N$ for random $x \in [1, M]$ and $M \leq N$. We conclude from his result that, for all primes N and for $x \in_R [1, N2^{-k}]$, the $(n/2 - k - (\log n)^2)$ least-significant bits of $P(x) \pmod N$ are statistically random in the sense of Santa and Vazirani (1984). This may be seen as evidence that the RSA-generator is perfect and that our question $\mathbf{Q}$ is sound.

The random variables Y, Z are called *statistically indistinguishable* within ε if $\sum_{\alpha} |\text{prob}[Y = \alpha] - \text{prob}[Z = \alpha]| \leq \varepsilon$. In this case and if Y ranges uniformly over some finite set S, then Z is called *statistically random within* ε in S. An ensemble

$(X_n)_{n\in\mathbb{N}}$ is called *statistically random* in the sequence $(S_n)_{n\in\mathbb{N}}$ of sets S_n if X_n is statistically random within $\varepsilon(n)$ in S_n where $\varepsilon(n)$ has the property that $\varepsilon(n) = O(n^{-t})$ for all $t > 0$. It can be easily seen that if the ensemble $(X_n)_{n\in\mathbb{N}}$ is statistically random in $(S_n)_{n\in\mathbb{N}}$ it is also pseudorandom in $(S_n)_{n\in\mathbb{N}}$ but the converse does not hold in general. For this we note that if the random variables Y and Z are statistically indistinguishable within ε, then for any function f the random variables $f(Y)$, $f(Z)$ too are statistically indistinguishable within ε. To prove the claim we apply this observation to $Y = X_n$ to the random variable Z that ranges uniformly on S_n, and to the function $f = C_n$ where $(C_n)_{n\in\mathbb{N}}$ is any poly-size circuit family.

For a polynomial $P \in \mathbb{Z}[x]$ we let $\deg_N(P)$, the *degree* of P on $\mathbb{Z}_N$, be the minimal degree of any polynomial $\bar{P} \in \mathbb{Z}[x]$ with the property that $P(x) = \bar{P}(x) \bmod N$ for all $x \in \mathbb{Z}$.

Theorem 4.1. *Let N be any prime that is n bits long and let $P \in \mathbb{Z}[x]$ be any polynomial with $\deg_N(P) \geq 2$. Then for all k, $m > 0$ and $x \in_R [1, N2^{-k}]$ we have that $[P(x)]_N \bmod 2^m$ is statistically random within ε in $[1, 2^m]$ where $\varepsilon = O(\deg_N(P)n^2 2^{-n/2+k+m})$.*

In particular it follows with $m = (n/2 - k - (\log n)^2)$ that if $e \geq 3$, N is a prime with $e \neq 0 \pmod{N-1}$, then the $(n/2 - k - (\log n)^2)$ least-significant bits of $x^e \pmod N$ are statistically random within ε for $\varepsilon = O(\deg_N(P)n^2 n^{-\log n})$. Note that such ε is for all $t > 0$ smaller than n^{-t} for all sufficiently large n.

We now state Niederreiter's result. Theorem 2 in Niederreiter (1988) asserts that Lemma 4.2 below holds for all polynomials $P \in \mathbb{Z}[x]$ that act as permutations on $\mathbb{Z}_N$. The proof given by Niederreiter proves Lemma 4.2 even for all polynomials $P \in \mathbb{Z}[x]$ that are nonlinear on $\mathbb{Z}_N$ since Niederreiter's proof does not use the permutation property. Let CH_I denote the characteristic function of the interval $I \subset [1, N]$.

Lemma 4.2 (Niederreiter, 1988, Theorem 2). *There exists an absolute constant $c > 0$ such that, for all primes N, all $P \in \mathbb{Z}[x]$ that are nonlinear on $\mathbb{Z}_N$, all $0 < M \leq N$, and all intervals $I \subset [1, N]$,*

$$\left|\left(\frac{1}{M}\sum_{1\leq x\leq M} CH_I(P(x) \bmod N)\right) - \frac{|I|}{N}\right| < c\frac{\deg_N(P)}{M}N^{1/2}(\log_2 N)^2.$$

If $M > N^{1/2}2^{(\log n)^2}$, then the expression in Lemma 4.2 is less than n^{-t} for all $t > 0$ and all sufficiently large n. From this we can show that the $(n/2 - k - (\log n)^2)$ least-significant bits of $P(x) \pmod N$ are statistically random for $x \in_R [1, N2^{-k}]$. Note that the assertion of Lemma 4.2 does not hold for linear polynomials P. We are now ready to prove Theorem 4.1.

Proof of Theorem 4.1. For simplicity we write $[z]$ for $[z]_N$. Given that N is odd any integer $b \pmod{2^m}$ can be uniquely written $b = -iN \pmod{2^m}$ with $i \in [0, 2^m - 1]$ and we have the following equivalence for $y \in [1, N]$:

$$y \in [iN2^{-m}, (i+1)N2^{-m}] \quad \Leftrightarrow \quad [2^m y] = -iN \pmod{2^m}.$$

The implication from the left to the right is obvious. It also proves the inverse implication.

We apply Lemma 4.2 with $M = N2^{-k}$, $I = [iN2^{-m}, (i+1)N2^{-m}]$ and with the polynomial $2^{-m}P \pmod N$. Then the above equivalence shows that, for all $i \in [0, 2^m - 1]$,

$$\left| \frac{\#\{x \in [1, N2^{-k}] | [P(x)] = -iN \pmod{2^m}\}}{N2^{-k}} - \frac{1}{2^m} \right| \leq \frac{c \cdot \deg_N(P)}{N2^{-k}} N^{1/2} n^2.$$

By summing up over all i we obtain the upper bound $c\ deg_N(P) n^2 2^{-n/2+k+m}$. □

Composite Moduli N. Recently, Niederreiter has extended Lemma 4.2 from prime moduli N to composite moduli $N = q_1 q_2$ that are products of two distinct primes q_1, q_2. In the extended Lemma 4.2 the degree $\deg_N P$ is replaced by the product $\deg_{q_1} P \cdot \deg_{q_2} P$. With this substitution, Theorem 4.1 also extends to RSA-moduli.

We next extend Theorem 4.1 from the least-significant bits of $[P(x)]_N$ to the leading bits. Since the leading bits of $[P(x)]_N$ are biased according to the leading bits of N we consider the distribution of $\lfloor [P(x)]_N 2^m / N \rfloor$ instead.

Theorem 4.3. *Let P, N, n, k, m be as in Theorem 4.1. Then the following distributions are statistically indistinguishable within ε where $\varepsilon = O\ \deg_N(P)\ n^2 2^{-n/2+k+m}$):*

- $\lfloor [P(x)]_N 2^m / N \rfloor$ *for* $x \in_R [1, N2^{-k}]$.
- $x \in_R [0, 2^m - 1]$.

Proof. It follows from Theorem 4.1 that, for $x \in_R [1, N2^{-k}]$, the m least-significant bits of $[2^m P(x)]_N$ are statistically random within ε where $\varepsilon = O(\deg_N(P) n^2 2^{-n/2+k+m})$. This follows by applying Theorem 4.1 to the polynomial $2^m P \pmod N$. From the proof of Theorem 4.1 we have the following equivalence for $i \in [0, 2^m - 1]$:

$$\begin{aligned} [2^m P(x)]_N = -iN \pmod{2^m} &\Leftrightarrow [P(x)]_N \in [iN2^{-m}, (i+1)N2^{-m}] \\ &\Leftrightarrow \lfloor [P(x)]_N 2^m / N \rfloor = i. \end{aligned}$$

The claim follows from this equivalence and since the m least-significant bits of $[2^m P(x)]_N$ are statistically random within ε. □

Remark. From the proof of Theorem 4.3 we have the following equivalence which holds for all $\varepsilon > 0$, all $k, m \in \mathbb{N}$, all nonlinear $P \in \mathbb{Z}[x]$, and for $x \in_R [1, N2^{-k}]$:

$$\begin{aligned} &[[2^m P(x)]_N]_{2^m} \text{ is statistically random within } \varepsilon \text{ in } [1, 2^m] \\ &\quad \Leftrightarrow \lfloor [P(x)]_N 2^m / N \rfloor \text{ is statistically random within } \varepsilon \text{ in } [0, 2^m - 1]. \end{aligned}$$

Theorems 4.1 and 4.3 show that the distribution of $[P(x)]_N$ for $x \in_R [1, N2^{-k}]$ approaches, in a weak sense, the uniform distribution on $[1, N]$. This shows that our question **Q** is sound from the statistical point of view. We now pose question $\mathbf{Q}_2$ which adapts question **Q** to prime moduli N. We let N_n denote a prime that is n bits long, $2^{n-1} < N_n < 2^n$.

Q_2. *Let $d \geq 2$ be an integer and let $k_d: \mathbb{N} \to \mathbb{N}$ be a function depending on d. Is it true that for all sequences $(N_n)_{n \in \mathbb{N}}$ of n bit primes and $(P_n)_{n \in \mathbb{N}}$ of polynomials with $\deg_{N_n}(P_n) = d$ the number $[[P_n(x)]_{N_n}]_{2^{k_d(n)}}$ is pseudorandom in $[1, 2^{k_d(n)}]$ for $x \in_R [1, N_n 2^{-k_d(n)}]$?*

We see from Theorem 4.1 that Q_2 holds with $k_d(n) = \lceil n(\frac{1}{4} - \delta) \rceil$ for any fixed $d \geq 2$ and $\delta > 0$. For such function k_d and $x \in_R [1, N_n 2^{-k_d(n)}]$ the ensemble $[P_n(x)]_{N_n}$ mod $2^{k_d(n)}$ is even statistically random in $[1, 2^{k_d(n)}]$. If Q_2 holds with $k_d(n) > n/2$ this yields a perfect RNG. If Q_2 holds for the function k_d we must have $k_d(n) < n(1 - 1/d)$; this is because $k_d(n) \geq n(1 - 1/d)$ implies $x^d \leq N_n$ for all $x \in [1, N_n 2^{-k_d(n)}]$. It is open whether Q_2 holds for $k_d(n) = \lfloor n(1 - 2/d) \rfloor$; this would yield a very efficient RNG. We next give an equivalent formulation for Q_2 in which the interval $[1, N_n 2^{-k_d(n)}]$ is replaced by an arbitrary pattern for the $k_d(n)$ least-significant bits of x.

Theorem 4.4. *Q_2 holds for $k_d: \mathbb{N} \to \mathbb{N}$ iff, for all sequences $(N_n)_{n \in \mathbb{N}}$ of primes and $(P_n)_{n \in \mathbb{N}}$ of polynomials with $\deg_{N_n}(P_n) = d$ and for all sequences $(y_n)_{n \in \mathbb{N}}$ with $y_n \in [1, 2^{k_d(n)}]$, we have that $[[P_n(x)]_{N_n}]_{2^{k_d(n)}}$ is pseudorandom in $[1, 2^{k_d(n)}]$ for $x \in_R [1, N_n] \cap (y_n + 2^{k_d(n)}\mathbb{Z})$.*

Proof. We abbreviate $k_d(n)$ by k. We know from the proof of Theorem 4.1 that the linear transformation $x \mapsto 2^k x \pmod{N_n}$ yields a bijection between the integer interval $I_i := [iN_n 2^{-k}, (i+1)N_n 2^{-k}]$ and the set $A_i := [1, N_n] \cap (-iN_n + 2^k\mathbb{Z})$. Let z_i denote the minimal integer in I_i. Then the linear transformation $x \mapsto 2^k(x + z_i - 1) \pmod{N_n}$ maps the interval $[1, N_n 2^{-k}]$ into the set A_i. The image of $[1, N_n 2^{-k}]$ covers the set A_i except possibly the maximal element of A_i. The exception occurs if $[1, N_n 2^{-k}]$ has fewer elements than I_i. Note that the interval $[1, N_n 2^{-k}]$ has $\lfloor N_n 2^{-k} \rfloor$ elements and I_i may have either $\lfloor N_n 2^{-k} \rfloor$ or $\lfloor N_n 2^{-k} \rfloor + 1$ elements.

In order to prove the theorem we set $i := y_n$ and we note that every polynomial $P_n \in \mathbb{Z}[x]$ corresponds to some polynomial $\bar{P}_n \in \mathbb{Z}[x]$ such that $P_n(x) = \bar{P}_n(2^k(x + z_i - 1)) \pmod{N_n}$. Conversely there exists for every $\bar{P}_n$ a corresponding P_n and these polynomials have the same degree when considered modulo N_n. Now if Q_2 holds for P_n, then the second statement of Theorem 4.4 holds for $\bar{P}_n$ and conversely. □

Corollary 4.5. *If Q_2 holds for the function $k_d: \mathbb{N} \to \mathbb{N}$, then for all sequences $(N_n)_{n \in \mathbb{N}}$ of primes and $(P_n)_{n \in \mathbb{N}}$ of polynomials with $\deg_{N_n}(P_n) = d$ we have that $([x]_{2^{k_d(n)}}, [[P_n(x)]_{N_n}]_{2^{k_d(n)}})$ is pseudorandom in $[1, 2^{k_d(n)}]^2$ for $x \in_R [1, N_n]$.*

Proof. Suppose that Q_2 holds for the function k_d. Then $[[P_n(x)]_{N_n}]_{2^{k_d(n)}}$ is pseudorandom in $[1, 2^{k_d(n)}]$ for $x \in_R [1, N_n]$. Moreover $[x]_{2^{k_d(n)}}$ is obviously pseudorandom in $[1, 2^{k_d(n)}]$ for $x \in_R [1, N_n]$. Now the claim follows from Theorem 2.1 (Yao, 1982) provided that $[x]_{2^{k_d(n)}}$ is unpredictable when given $[[P_n(x)]_{N_n}]_{2^{k_d(n)}}$. In order to prove this unpredictability we consider any two distinct sequences of integers $(y_n)_{n \in \mathbb{N}}$. By Theorem 4.4 the corresponding ensembles

$$[[P_n(x)]_{N_n}]_{2^{k_d(n)}} \qquad \text{for} \quad x \in_R [1, N_n] \cap (y_n + 2^{k_d(n)}\mathbb{Z})$$

are polynomially indistinguishable. This shows that the bits of x for which the two

integers y_n differ, cannot be predicted when given $[[P_n(x)]_{N_n}]_{2^{k_d(n)}}$. Since this holds for any two sequences $(y_n)_{n \in \mathbb{N}}$ it follows that $[x]_{2^{k_d(n)}}$ is unpredictable when given $[[P_n(x)]_{N_n}]_{2^{k_d(n)}}$. This proves the claim as shown above. □

Example of a Perfect RNG. Let $n = 0 \pmod 4$, let $d = 8$, let $P(x) = x^8$, and let $h(n)$ be polynomial.

input $x_1 \in_R [1, N_n 2^{-3n/4}]$, N_n an n bit prime.
for $i = 1, \ldots, h(n)$ do
$z_i := [x_i^8]_{N_n}$, $x_{i+1} := \lfloor [z_i]_{2^{3n/4}} 2^{-n/2} \rfloor + 1$
output $\mathrm{Out}(x_i) := z_i[n/2$, the $n/2$ least-significant bits of z_i, for $i = 1, \ldots, h(n)$.

Corollary 4.6. *If* $\mathbf{Q}_2$ *holds with* $k_d(n) = \lfloor n(1 - 2/d) \rfloor$, *then the above* RNG *is perfect.*

The proof of perfectness for this generator follows the proof of Theorem 5.1 and is given subsequent to this proof.

This generator outputs $n/2$ bits per iteration at a cost of about 1.25 modular multiplications. In each iteration we compute x_i^2, x_i^4, and $x_i^8 \pmod{N_n}$. Since $x_i^4 \leq N_n$ only the computation of $x_i^8 \pmod{N_n}$ requires a full modular multiplication. The other multiplications are with small numbers. The security of this generator does not rely on the assumption that the modulus N_n is difficult to factor. Generators with prime moduli are particularly suited for some applications. Everybody can use this generator with the same modulus N and no trusted authority is required for generating N.

It is open whether the SPG of Section 3 is perfect for arbitrary prime moduli N that are n bits long and for some function $k_e(n)$ with $1 \leq k_e(n) \leq n(1 - 2/e)$. This generator may be perfect even though $\mathbf{Q}_1$ does not hold for prime moduli. Since the perfectness of this generator does not follow from $\mathbf{Q}_2$ it requires a more general hypothesis. This generator, if perfect, could be practical for smaller moduli N. For a 256-bit modulus N, $e = 4$, and $k = n/2$ we could generate 128 pseudorandom bits at the cost of about 1.25 modular multiplications.

5. Perfectness of Sequential and Parallel Generators

Theorem 5.1. *Suppose that* $\mathbf{Q}_1$ *holds for* $k(n)$. *Then for random* $N \in_R S_{n,e}$, $x \in_R [1, N2^{-k(n)}]$, *and polynomial* $h = h(n)$ *the output* $\mathrm{SPG}_{h,e}(x, N)$ *of the sequential generator is pseudorandom.*

Proof. We abbreviate $h = h(n)$, $k = k(n)$. We transform any statistical test C that ε_n-rejects $(N, \mathrm{SPG}_{h,e}(x, N))$ into a test C_i that ε_n/h-rejects $(N, [x^e]_N)$. Let C be a statistical test that takes inputs of the form

$$N y_1 \cdots y_h \in S_{n,e} \times \{0, 1\}^{kh}.$$

We consider the "hybrid" random variables X_i, $i = 0, \ldots, h$, ranging over such inputs:

$$X_i = N r_1 \cdots r_i \, \mathrm{Out}(x_{i+1}) \cdots \mathrm{Out}(x_h),$$

where $N \in_R S_{n,e}$, $r_1 \cdots r_i \in_R \{0,1\}^{ki}$ and $\mathrm{Out}(x_{i+1}) \cdots \mathrm{Out}(x_h)$ is produced by picking a random $z \in_R [1, N]$ and by applying the SPG with input N and $x_{i+1} := \lfloor z2^{-k} \rfloor + 1$.

Let p_i be the probability that C outputs 1 with input X_i. If C ε_n-rejects $(N, \mathrm{SPG}_{h,e}(x, N))$ for $N \in_R S_{n,e}$, $x \in_R [1, N2^{-k}]$, then we have $|p_0 - p_h| \geq \varepsilon_n$. Therefore it is sufficient to derive from C a probabilistic test C_i that $|p_{i-1} - p_i|$-rejects $(N, [x^e]_N)$ for random $N \in_R S_{n,e}$, and $x \in_R [1, N2^{-k}]$. The claim follows since we have $|p_{i-1} - p_i| \geq \varepsilon_n/h$ for some $i \leq h$. We let C_i take inputs of the form $(N, z) \in S_{n,e} \times [1, N]$. C_i partitions z into $z[k$ and $x_{i+1} := \lfloor z2^{-k} \rfloor + 1$, generates random $r_1, \ldots, r_{i-1}$ by internal coin tosses, and applies C with input $Nr_1 \cdots r_{i-1} z[k\, \mathrm{Out}(x_{i+1}) \cdots \mathrm{Out}(x_h)$. With random input $N \in_R S_{n,e}, z \in_R [1, N], z[k$ is a random number that is independent of x_{i+1}, and thus C_i outputs 1 with probability p_i. (Here we neglect a negligible dependence between $z[k$ and x_{i+1} that arises from the case where $\lfloor z2^{-k} \rfloor = \lfloor N2^{-k} \rfloor$ and which is due to the condition that $z \leq N$. This dependence is negligible since this case has probability $\leq 2^k/N$.) With input $N \in_R S_{n,e}$, $z := [x^e]_N$ for $N \in_R S_{n,e}$, $x \in_R [1, N2^{-k}]$ the distribution of $(z[k, x_{i+1})$ is the same as the distribution of $(\mathrm{Out}(x_i), x_{i+1})$ within X_{i-1}, and thus C_i outputs 1 with probability p_{i-1}. □

Proof of Corollary 4.6. This proof follows the proof of Theorem 5.1 with the following changes. Now we have $k = k(n) = n/2$ and the modulus N_n is constant whereas $N \in_R S_{n,e}$ is random in Theorem 5.1. Now we compute $x_{i+1} := \lfloor [z]_{2^{3n/4}} 2^{-n/2} \rfloor + 1$ whereas we have $x_{i+1} := \lfloor z2^{-k} \rfloor + 1$ in the proof of Theorem 5.1. With these changes define the random variables $X_i = N_n r_1 \cdots r_i \mathrm{Out}(x_{i+1}) \cdots \mathrm{Out}(x_k)$ as in the proof of Theorem 5.1 but generate $\mathrm{Out}(x_{i+1}) \cdots \mathrm{Out}(x_k)$ from x_{i+1} by applying the generator of Corollary 4.6. We can transform any statistical test C that ε_n-rejects X_0 into a test C_i that ε_n/n-rejects $[[x^8]_{N_n}]_{2^{3n/4}}$ for $x \in_R [1, N_n 2^{-3n/4}]$, a contradiction to $\mathbf{Q}_2$. On input $z \in [1, N]$ the test C_i applies C to the random variable

$$N_n r_1 \cdots r_{i-1}\, z[k\, \mathrm{Out}(x_{i+1}) \cdots \mathrm{Out}(x_h).$$

In order to simulate C with input X_i (resp. X_{i-1}) apply the test C_i with $z \in_R [1, N_n]$ (resp. with $z = [x^8]_{N_n}$ for $x \in_R [1, N_n 2^{-3n/4}]$). □

From Theorems 3.1 and 5.1 we have the following corollary.

Corollary 5.2. *If the RSA-scheme is secure, $k(n) = O(\log n)$, and $h(n)$ is polynomial, then the sequential polynomial generator is perfect.*

Remark. It is tempting to generate pseudorandom bits according to the recursion $x_{i+1} := x_i^e \pmod N$, $i = 1, 2, \ldots$, and to output the $k(n)$ least-significant bits of x_i. This "incestuous" generator is more difficult to justfy by a theorem similar to Theorem 4.1. The period of the sequence x_i, $i = 1, 2, \ldots$, raises further doubts concerning this generator. The period is the order of e in $\mathbb{Z}^*_{\varphi(N)}$. In general it will be a large factor of $\varphi\varphi(N)$ and will be much larger than $\sqrt{N}$ which is the average period of a random recursion in $\mathbb{Z}_N$. It is conceivable that the number $\varphi\varphi(N)$ somewhat affects the output distribution of the generator and not only its period.

The "incestuous" generator is less efficient than our generator in case that $k(n)$ is sufficiently large. It requires more iterations than the SPG to obtain the same expansion factor. In addition computing $x_i^e \pmod N$ for a full size x_i and $e \geq 3$ requires at least two modular multiplications whereas an iteration for our generator only costs about one modular multiplication for large enough $k(n)$. For example, we consider the case $e = 7$, $k(n) = n\frac{5}{7}$.

Example. Suppose that the answer to $\mathbf{Q}_1$ is "yes" for $e = 7$ and $k(n) = n\frac{5}{7}$. Let $N \in S_{n,7}$ with $n = 512$. Then for the above RNG $\mathrm{Out}(x_i)$ consists of the 365 least-significant bits of $x_i^7 \pmod N$ and x_{i+1} is the number corresponding to the 146 most-significant bits of $x_i^7 \pmod N$. For $x_i \in [1, 2^{146}]$ we compute $x_i^7 \pmod N$ by computing $x_i^2, x_i^4, x_i^7 = x_i \cdot x_i^2 \cdot x_i^4$. Since $x_i^3 \leq 2^{512 \cdot 6/7}$ the cost of the multiplication $x_i^3 \cdot x_i^4$ is about $\frac{6}{7}$ times the cost of one full modular multiplication. The cost per iteration corresponds to one full modular multiplication.

Efficient Public Key Encoding and Decoding. We can use the above RNGs to generate a one-time-pad for message encoding. When given the seed x_1 of the one-time-pad, encoding and decoding can be done at a speed of about $k(n)$ bits per multiplication modulo N. A public key coding scheme as, e.g., RSA can be used to encode and to decode the seed x_1.

The Parallel Polynomial Generator. The *parallel polynomial generator* (PPG) generates from random seed $x \in [1, N2^{-k(n)}]$ a tree with root x and outdegree s with $s = \lfloor n/(n - k(n)) \rfloor$. The nodes of this *iteration tree* are pseudorandom numbers in $[1, N2^{-k(n)}]$ that are represented by bit strings of length $l = n - k(n)$. We consider the case that $k(n) > n/2$ and $s \geq 2$.

The *successors* $y(1), \ldots, y(s)$ of a node y and the *output string* $\mathrm{Out}(y)$ of node y are defined as follows. Let $b_1, \ldots, b_n$ be the bits of the binary representation of $y^e \bmod N$, with b_1 being the most-significant bit, i.e.,

$$\sum_{i=1}^{n} b_i 2^{n-i} = y^e \mod N.$$

We partition the sl most-significant bits into s blocks with l bits in each block. The corresponding numbers

$$y(j) := 1 + \sum_{i=1}^{l} b_{(j-1)l+i} 2^{l-i} \qquad \text{for} \quad j = 1, \ldots, s$$

are the successors of node y in the iteration tree. The *output* $\mathrm{Out}(y)$ at node y consists of the remaining low-order bits of $y^e \bmod N$,

$$\mathrm{Out}(y) = b_{sl+1} \cdots b_n.$$

For convenience we denote the nodes on level h of the iteration tree as $x(j_1, \ldots, j_h)$; $x(j_1, \ldots, j_{h-1})$ is the direct predecessor of $x(j_1, \ldots, j_h)$ and j_h ranges from 1 to s_{h-1}.

The parallel polynomial generator can be illustrated by the infinite tree shown in Fig. 2.

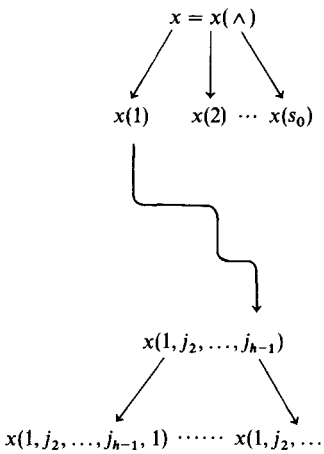

Fig. 2. The parallel polynomial generator (PPG).

We define the *output* $\mathrm{PPG}_{h,e}(x, N)$ of the PPG with seed x as the concatenation of all bit strings $\mathrm{Out}(x(j_1, \ldots, j_i))$ on levels i with $0 \leq i \leq h$, with respect to any efficient enumeration order, as, e.g., preorder traversal, postorder traversal, inorder traversal, or enumeration by levels.

The argument of Goldreich *et al.* (1986) extends Theorem 5.1 to the parallel generator provided that we process at most polynomially many nodes in the iteration tree. This yields the following corollary.

Corollary 5.3. *Suppose the answer to* $\mathbf{Q}_1$ *is yes for* $k(n) > n/2$. *Then for random* $N \in_R S_{n,e}$, $x \in_R [1, N2^{-k(n)}]$ *the output* $\mathrm{PPG}_{h,e}(x, N)$ *of the parallel generator is pseudorandom provided that the output has polynomial length.*

Parallel Mode for Arbitrary Perfect RNGs. The method of parallelization applies to every perfect RNG. The parallel variant of the generator associates an iteration tree to a random seed. For example, suppose the sequential generator G_n: $\{0, 1\}^n \to \{0, 1\}^{3n}$ stretches random bit strings of length n into pseudorandom bit strings of length $3n$. We construct from random seed $w \in_R \{0, 1\}^n$ a binary iteration tree with nodes in $\{0, 1\}^n$ and with root ω. Construct the two successors $y(1)$, $y(2)$ and the output $\mathrm{Out}(y)$ of node y by partitioning $G_n(y) \in \{0, 1\}^{3n}$ into three substrings of length n, $G_n(y) = y(1)y(2)\,\mathrm{Out}(y)$. The output of the parallel generator is the concatenation of $\mathrm{Out}(y)$ for all nodes of depth $\leq h$. This output is pseudorandom provided that its length is polynomial and that $(G_n)_{n \in \mathbb{N}}$ is perfect.

Fast Retrieval for the Parallel Generator. There is an efficient straightforward way to retrieve substrings of the output of the parallel generator. Consider, for example, the case of a complete iteration tree of outdegree 2. Level h of the tree has 2^h nodes. Suppose that the nodes of the tree are enumerated in preorder traversal. To retrieve node y we follow the path from the root to y. This requires processing and storage of at most h nodes and can be done at the costs of about h evaluations of G_n.

Acknowledgment

The authors wish to thank A. K. Lenstra and A. Shamir for discussions during this work. H. Niederreiter brought Lemma 4.2 to our attention.

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 J. Comput.*, **17** (1988), 194–208.

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*, New York (1982); also *SIAM J. Comput.*, **13** (1984), 850–864.

Blum, L., Blum, M., and Shub, M.: A simple unpredictable pseudo-random number generator. *SIAM J. Comput.*, **15** (1986), 364–383.

Goldreich, O., Goldwasser, S., and Micali, S.: How to construct random functions. *Proceedings of the* 25th *IEEE Symposium on Foundations of Computer Science*, New York (1984); also *J. Assoc. Comput. Mach.*, **33** (1986), 792–807.

Knuth, D. E.: *The Art of Computer Programming*, Vol. 2, 2nd edn. Addison-Wesley, Reading, MA (1981).

Lenstra, A. K., Lenstra, H. W., Jr., Manasse, M. S., and Pollard, J. M.: The number field sieve. *Proceedings of the* 22nd *ACM Symposium on Theory of Computing*, Baltimore (1990).

Micali, S., and Schnorr, C. P.: Super-efficient perfect random number generators. *Proceedings Crypto* '88. Lecture Notes in Computer Science, Vol. 403. Springer-Verlag, Berlin (1988).

Niederreiter, H.: Statistical independence of nonlinear congruential pseudorandom numbers. *Mh. Math.*, **106** (1988), 149–159.

Niederreiter, H.: Private communication (1990).

Pollard, J.: Private communication (1988).

Santa, M., and Vazirani, U. V.: Generating quasi-random sequences from slightly random sources. *Proceedings of the* 25th *Annual IEEE Symposium on Foundations of Computer Science*, Singer Island (1984), 434–440.

Solovay, R., and Strassen, V.: A fast Monte Carlo test for primality. *SIAM J. Comput.*, **6** (1977), 84–85, erratum **7** (1978), 118.

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