SIAM J. COMPUT.
Vol. 15, No. 2, May 1986

003

# A SIMPLE UNPREDICTABLE PSEUDO-RANDOM NUMBER GENERATOR*

L. BLUM†, M. BLUM‡ AND M. SHUB§

**Abstract.** Two closely-related pseudo-random sequence generators are presented: The $1/P$ *generator*, with input $P$ a prime, outputs the quotient digits obtained on dividing 1 by $P$. The $x^2 \bmod N$ *generator* with inputs $N$, $x_0$ (where $N = P \cdot Q$ is a product of distinct primes, each congruent to 3 mod 4, and $x_0$ is a quadratic residue mod $N$), outputs $b_0 b_1 b_2 \cdots$ where $b_i = \text{parity}(x_i)$ and $x_{i+1} = x_i^2 \bmod N$.

From short seeds each generator efficiently produces long well-distributed sequences. Moreover, both generators have computationally hard problems at their core. The first generator's sequences, however, are *completely predictable* (from any small segment of $2|P|+1$ consecutive digits one can infer the "seed," $P$, and continue the sequence backwards and forwards), whereas the second, under a certain intractability assumption, is *unpredictable* in a precise sense. The second generator has additional interesting properties: from knowledge of $x_0$ and $N$ but *not* $P$ or $Q$, one can generate the sequence forwards, but, under the above-mentioned intractability assumption, one can *not* generate the sequence backwards. From the additional knowledge of $P$ and $Q$, one *can* generate the sequence backwards; one can even "jump" about from any point in the sequence to any other. Because of these properties, the $x^2 \bmod N$ *generator* promises many interesting applications, e.g., to public-key cryptography. To use these generators in practice, an analysis is needed of various properties of these sequences such as their periods. This analysis is begun here.

**Key words.** random, pseudo-random, Monte Carlo, computational complexity, secure transactions, public-key encryption, cryptography, one-time pad, Jacobi symbol, quadratic residuacity

What do we want from a pseudo-random sequence generator? Ideally, we would like a pseudo-random sequence generator to quickly produce, from short seeds, long sequences (of bits) that appear in every way to be generated by successive flips of a fair coin.

Certainly, the idea of a (fast) deterministic mechanism producing such non-deterministic behavior seems contradictory: by observing its outcome over time, we could in principle eventually detect the determinism and simulate such a generator.

The resolution [Knuth], usually, is to require of such generators only that the sequences they produce pass certain standard statistical tests (e.g., in the long run, the frequency of 0's and 1's occurring in such a sequence should be nearly the same, and the 0's and 1's should be "well-mixed").

However, the usual statistical tests do not capture enough. An important property of sequences of coin tosses is their unpredictability. Pseudo-random sequences should be unpredictable to computers with feasible resources. We say that a pseudo-random sequence generator is *polynomial-time unpredictable* (unpredictable to the right, unpredictable to the left) [Shamir], [Blum–Micali] if and only if for every finite initial segment of sequence that has been produced by such a generator, but with any element (the rightmost element, the leftmost element) deleted from that segment, a probabilistic

* Received by the editors September 7, 1982, and in final revised form August 15, 1983. A preliminary version of this paper was presented at Crypto 82.

† Department of Mathematics and Computer Science, Mills College, Oakland, California 94613, and Department of Mathematics, University of California at Berkeley, Berkeley, California 94720. This work was supported in part by the Letts-Villard Chair, Mills College.

‡ Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, California 94720. This work was supported in part by the National Science Foundation under grant MCS 82-04506.

§ IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598, and City University of New York, New York, New York 10036. This work was supported in part by the National Science Foundation under grant MCS 82-01267.

Turing machine can, roughly speaking, do no better in guessing in polynomial time (polynomial in the length of the "seed," cf. § 2) what the missing element is than by flipping a fair coin.

**1. Two pseudo-random sequence generators.** In this paper, two pseudo-random sequence generators are defined and their properties discussed. These are called:

(1) the $1/P$ *generator,*

(2) the $x^2 \bmod N$ *generator.*

The two generators are closely related. For example: *From short seeds, each quickly generates long well-distributed sequences. Both generators contain hard problems at their core* (the discrete logarithm problem and the quadratic residuacity problem, respectively). *But only the second is "unpredictable"—assuming a certain intractibility hypothesis.*

More specifically, THEOREM 2, Problem 4 (§ 6). *Any sequence produced by the* $1/P$ *generator is completely predictable; that is, given a small segment of the sequence, one can quickly infer the "seed" and efficiently extend the given segment backwards and forwards.*

On the other hand, THEOREM 4 (§ 7). *Modulo the quadratic residuacity assumption, the* $x^2 \bmod N$ *generator is polynomial-time unpredictable to the left. We say, for reasons pointed out in the applications* (§ 10), *that the sequences it generates are cryptographically secure.*

The $1/P$ generator has been well studied in the history of number theory [Dickson] and as a pseudo-random number generator [Knuth]. Our results concerning its strong inference properties, we believe, are new and surprising.

The $x^2 \bmod N$ generator is an outgrowth of the coin-flipping protocol of [Blum]. Its strong security properties derive from complexity based number theoretic assumptions and arguments [Blum], [Goldwasser–Micali], [Yao]. Our investigation reveals additional useful properties of this generator: e.g., from knowledge of the (secret) factorization of $N$, one can generate the sequence backwards; from additional information about $N$, one can even random access the sequence. Our number-theoretic analyses also provide tools for determining the lengths of periods of the generated sequences.

Both generators have applications. The $1/P$ generator has applications to the generation of generalized de Bruijn (i.e., maximum-length shift-register) sequences. The $x^2 \bmod N$ generator has applications to public-key cryptography.

The two generators are presented together so that each one's properties help to illuminate the other's.

**2. Notation and definitions.** In this paper, the underlying models of computation are Turing machines [Hopcroft and Ullman]. *Probabilistic procedures* are effective procedures (Turing machines) that can toss a fair coin (at a cost of 1 step per toss) to produce truly random bits during their computation. (*Probabilistic*) *polynomial-time procedures* halt in (worst-case) time $poly(n)$, where *poly* denotes a polynomial, and $n$ is the input *length.*

The *base,* $b$, will always be an integer $>1$. For any positive integer, $N$, let $|N|_b = \lfloor 1 + \log_b N \rfloor$ be the *length of* $N$ when $N$ is expanded base $b$, and let $|N| = |N|_2$. We also let $n = |N|$ so $N = O(2^n)$.

For $\Sigma = \{0, 1, \cdots, b-1\}$, let $\Sigma^*$ be the set of *finite sequences* of elements of $\Sigma$, and let $\Sigma^\infty$ be the set of (one-sided) *infinite sequences* of elements of $\Sigma$.

For $x \in \Sigma^*$, let $|x|$ be the *length of* $x$, and for integers $k \geqq 0$, let $\Sigma^k = \{x \in \Sigma^* \mid |x| = k\}$. For $x \in \Sigma^\infty$, and for integers $k \geqq 0$, let $x^k$ be the *initial segment of* $x$ of length $k$, and $x_k$ be the $k$th coordinate of $x$ where $x_0$ is the initial coordinate of $x$.

DEFINITION. Let $\mathbf{N}$ be a set of positive integers, the *parameter values*, and for each $N \in \mathbf{N}$, let $X_N \subset \{0,1\}^n$ be a set of *seeds* (recall $n = |N|$). The set $X = \{(N, x) | N \in \mathbf{N}, x \in X_N\}$ is called a *seed domain.*

We can, and sometimes do, think of $X_N$ as a subset of $X$ by identifying seed $x \in X_N$ with "seed" $(N, x) \in X$. With this identification, $X$ can be thought of as the disjoint union $\cup_{N \in \mathbf{N}} X_N$. The point of view should be clear from context.

DEFINITION. Let $X^n = \{(N, x) | N \in \mathbf{N}, |N| = n, \text{ and } x \in X_N\}$ be the set of *seeds of length n.* Suppose for all sufficiently large integers $n$, $\mu_n$ is a probability distribution on $X^n$. Then $U = \{\mu_n\}$ is an *accessible probability distribution on* $X$ if there is a polynomial *poly* and a probabilistic *poly*$(n)$-time procedure that for each sufficiently large input, $n$, outputs an element of $X^n$ according to $\mu_n$ *with negligible error*, i.e., it outputs an element of a set containing $X^n$ according to $\mu'_n$ ($\mu'_n$ = a probability distribution on the set containing $X^n$) where, for all $t$, for all sufficiently large $n$, $\sum_{(N,x) \in X^n} |\mu_n(N, x) - \mu'_n(N, x)| < 1/n^t$.

A pair $(X, U)$, where $X$ is a seed domain and $U$ is an accessible probability distribution on $X$, is called a *seed space.* We simply let $X$ denote the seed space when the underlying distribution is clear.

Now, let $\Sigma = \{0, 1, \cdots, b-1\}$.

DEFINITION. A (*base b*) *pseudo-random sequence generator G on seed space X* is an effective map $G: X \to \Sigma^\infty$ such that for each integer $s \geqq 0$, there is an integer $t \geqq 0$ such that for all $(N, x) \in X$ with $\mu_n(N, x) \neq 0$, $[G(N, x)]^{n^s}$, the initial segment of $G(N, x)$ of length $n^s$, is output in time $O(n^t)$. (Thus, from short "seeds" (i.e., of "length" $n$), that are produced using at most poly$(n)$ truly random bits, $G$ generates long sequences (i.e., of length $n^s$), in polynomial time.) $G(N, x)$ is called the *pseudo-random sequence generated by G* with *input* or *seed* $(N, x)$.

*Remark.* If $\Sigma$ represents a set of "observable states" for elements of seed space $X$, then the sequence $G(N, x)$ might represent the observed states through which seed $x$ passes (at times $0, 1, 2, \cdots$) resulting from some underlying transformation of $X$ into itself. This point of view motivates the following more structured (and more restrictive) formulation of a pseudo-random sequence generator.

DEFINITION. A *transformation* $T$ on seed space $X$ is a poly-time effective map $T: X \to X$ such that for all sufficiently large $n$, $T(X^n) \subset X^n$ and $T$ preserves $\mu_n$ (i.e., $\mu_n(A) = \mu_n(T^{-1}(A))$ for each $A \subset X^n$). For each seed $x \in X_N$, the sequence $x$, $Tx$, $T^2x, \cdots$ is called the *orbit* of $x$ under $T$. We sometimes write $x_k = T^k x$, so $x_0 = x$ and $x_{k+1} = T(x_k)$.

DEFINITION. A *partition* $B$ of seed space $X$ into states $\Sigma$ is a poly-time effective map $B: X \to \Sigma$.

The system $\langle X, T, B \rangle$, with $X$ a seedspace, $T$ a transformation on $X$, and $B$ a partition naturally defines a base $b$ pseudo-random sequence generator $G$ on $X$ where the $k$th coordinate, $[G(N, x)]_k = B(T^k x)$. Thus, if $x_0, x_1, x_2, \cdots$ is the orbit of $x$ under $T$, then $G(N, x) = b_0 b_1 \cdots$ where $b_k = B(x_k)$ is the state of $x$ at time $k$.

*Remark.* If $T$ is poly-time invertible on $X$, i.e., if $T^{-1}$ is defined and poly-time computable, we can, and sometimes do, think of $G$ mapping $X$ into the set of 2-sided infinite sequences on $\Sigma$.

In the next two sections we give examples of specific pseudo-random sequence generators. We use $x^2$ *mod N generator* to denote a particular type of pseudo-random sequence generator, whereas $x^2$ mod $N$ denotes the remainder upon dividing a specific integer $x^2$ by $N$. A similar distinction is made between the $1/P$ *generator* and the string of digits $1/P$.

Throughout this paper $x \bmod N$ denotes the least nonnegative integer remainder upon dividing $x$ by $N$ (rather than denoting the residue class mod $N$).

Recall that $Z_N^* = \{\text{integers } x | 0 < x < N \text{ and } \gcd(x, N) = 1\}$ is a multiplicative group of order $\varphi(N)$. If $P$ is prime, then $Z_P^* = \{1, 2, \cdots, P-1\}$ is cyclic. For each $N$, we consider $Z_N^* \subset \{0,1\}^n$ via the natural identification.

**3. The $1/P$ generator.** Fix an integer $b > 1$ and let $\Sigma = \{0, 1, \cdots, b-1\}$.

DEFINITION ($1/P$ *generator* (*base* $b$)). To define the *seed space*, let $\mathbf{N} = \{\text{integers } P > 1 \text{ relatively prime to } b\}$ be the *parameter values*, and let the *seed domain* $X$ be the disjoint union $\cup_{P \in \mathbf{N}} Z_P^*$. We can, and sometimes do, identify $X$ with the (dense) subset $\{r/P | P \in \mathbf{N}, r \in Z_P^*\}$ of the unit interval $[0, 1)$. Let $\mu_n$ be the distribution on $X^n$ given by $\mu_n(P, r) = u_n(P) \cdot v_P(r)$, where $u_n$ is the uniform probability distribution on $\{P \in \mathbf{N} \,||P|_b = n\}$ and $v_P$ is the uniform distribution on $Z_P^*$. Then $U = \{\mu_n\}$ is an accessible probability distribution on $X$.

Let $G: X \to \Sigma^\infty$ be defined by letting $G(r/P) = q_1 q_2 q_3 \cdots$ be the sequence of $b$-ary quotient digits that immediately follow the decimal point when $r/P \in X$ is expanded base $b$. (We note that the successive digits of $G(r/P)$ can be computed in $O(|b| \cdot |P|)$-time, and that the sequence $G(r/P)$ is periodic with period dividing $\varphi(P)$.) We call this pseudo-random sequence generator the $1/P$ *generator* (*base* $b$).

From the state space point of view, the $1/P$ generator (*base* $b$) is the pseudo-random sequence generator defined by the triple $\langle X, T, B \rangle$ where $X$ is the *seed space* defined above, the *transformation* $T: X \to X$ is defined for $x$ in $[0, 1)$ by $Tx = bx \bmod 1$ (equivalently $T(r) = br \bmod P$ for $r \in Z_P^*$, which is a permutation on $Z_P^*$), and the *partition* $B: X \to \Sigma = \{0, 1, 2, \cdots, b-1\}$ is defined for $x$ in $[0, 1)$ by $B(x) = \lfloor bx \rfloor$ (equivalently, $B(r) = \lfloor br/P \rfloor$ for $r \in Z_P^*$).

*Remark.* The $1/P$ *generator* (*base* 2) might be considered to be a discrete realization of the classical arithmetical model of a coin toss defined by the map $2x \bmod 1$ and partition $[0, \frac{1}{2}) \cup [\frac{1}{2}, 1)$ of the unit interval [Billingsley, Kac]. In § 6 we see that while a number of the "ergodic" like properties of the classical model are reflected in this discrete realization, the sequences produced are predictable.

*Example.* Let the base $b = 10$, and let $P = 7$ and $r = 1$. The pseudo-random sequence generated by the $1/P$ generator (*base* 10) with input $1/7$ is $142857142 \cdots$. Note that 10 is a primitive root mod 7 (i.e., a generator of the cyclic group $Z_7^*$) and that the period of this sequence is $7 - 1 = 6$ (see Theorem 1). From the state space point of view, the orbit of $1/7$ under $T$ is: $1/7, 3/7, 2/7, 6/7, 4/7, 5/7, 1/7, \cdots$, and so, $b_0 = 1$ (since $1/7 \in [1/10, 2/10)$), $b_1 = 4$ (since $3/7 \in [4/10, 5/10)$), $b_2 = 2$, $b_3 = 8$, $b_4 = 5, \cdots$.

**4. The generator $x^2 \bmod N$.**

DEFINITION [$x^2 \bmod N$ generator]. Let $\mathbf{N} = \{\text{integers } N | N = P \cdot Q, \text{ such that } P, Q \text{ are equal length } (|P| = |Q|) \text{ distinct primes} \equiv 3 \bmod 4\}$ be the set of *parameter values*. For $N \in \mathbf{N}$, let $X_N = \{x^2 \bmod N | x \in Z_N^*\}$ be the set of quadratic residues mod $N$. Let $X = \text{disjoint } \cup_{N \in \mathbf{N}} X_N$ be the *seed domain*.

For $(N, x) \in X^n$, let $\mu_n(N, x) = u_n(N) \cdot v_N(x)$, where $u_n$ is the uniform probability distribution on $\{N \in \mathbf{N} \,||N| = n\}$ and $v_N$ is the uniform distribution on $X_n$. Then $\{\mu_n\}$ is an *accessible probability distribution* on $X$ since

1. asymptotically, $1/(k \ln 2)$ of all $k$-bit numbers are prime and half the primes of any given lengths are $\equiv 3 \bmod 4$ (by de la Vallee Poussin's extension of the prime number theorem [Shanks]);

2. primality is decidable in polynomial time by (Monte-Carlo) probabilistic procedures [Strassen–Solovay], [Rabin '80] or, assuming the extended Riemann hypothesis, by a deterministic polynomial time procedure [Miller], and

3. gcds are computable in polynomial time, and $|Z_N^*|/|Z_N| \to 1$ as $n \to \infty$.

Let the *transformation* $T: X \to X$ be defined by $T(x) = x^2 \bmod N$ for $x \in X_N$. $T$ is a permutation on $X_N$ (see Lemma 1) and is computable in poly-time. Let the *partition* $B: X \to \{0, 1\}$ be defined by $B(x) =$ parity of $x$. $B$ is computable in poly-time. Then $\langle X, T, B\rangle$ defines a pseudo-random sequence generator (base 2) called the $x^2 \bmod N$ generator. Thus, with *inputs* $(N, x_0)$ the $x^2 \bmod N$ *generator outputs the pseudo-random sequence* of bits $b_0 b_1 \cdots$ obtained by setting $x_{i+1} = x_i^2 \bmod N$ and extracting the bit $b_i =$ parity $(x_i)$. Such sequences are periodic with period usually equal to $\lambda(\lambda(N))$ (see § 8 for the definition of $\lambda$ and clarification of "usually"). We also note that the equality $x_i = x_0^{2^i} \bmod N = x_0^{2^i \bmod \lambda(N)} \bmod N$ enables us to efficiently compute the $i$th sequence element, given $x_0$, $N$ and $\lambda(N)$, for $i > 0$. For $i < 0$, use $x_i = x_{i \bmod \lambda(\lambda(N))}$.

*Example.* Let $N = 7 \cdot 19 = 133$ and $x_0 = 4$. Then the sequence $x_0, x_1 = x_0^2 \bmod 133, \cdots$ has period 6, where $x_0, x_1, \cdots, x_5, \cdots = 4, 16, 123, 100, 25, 93, \cdots$. So $b_0 b_1 \ldots b_5 \ldots = 0\ 0\ 1\ 0\ 1\ 1 \cdots$. The latter string of $b$'s is the pseudo-random sequence generated by the $x^2 \bmod N$ generator with input (133, 4). Here, $\lambda(N) = 18$ and $\lambda(\lambda(N)) = 6$.

**5. The assumptions.** Our main results about unpredictability and cryptographic security follow from assumptions concerning the intractability of certain number-theoretic problems by probabilistic polynomial-time procedures. Stronger results would follow from stronger assumptions concerning the circuit size complexity of the number theoretic problems below. Such results would be desirable, for example, if we wished to assure that sequences produced by our generator appear random to hard-wired circuits.

1. *The discrete logarithm (index finding) problem.* Let $P$ be a prime. Let $b$ be a primitive root mod $P$ (i.e., a generator for $Z_P^*$). The function $f_{b,P}: Z_P^* \to Z_P^*$ defined by $f_{b,P}(x) = b^x \bmod P$ is a permutation of $Z_P^*$ that is computable in polynomial time. The *discrete logarithm (index finding) problem* with parameters $b$ and $P$ consists in finding for each $y$ in $Z_p^*$ the index $x$ in $Z_p^*$ such that $b^x \bmod P = y$. A (probabilistic) procedure $\mathbf{P}[b, P, y]$ solves the discrete logarithm problem if for all primes $P$, for all generators $b$ for $Z_P^*$, and for all $y$ in $Z_P^*$, $\mathbf{P}[b, P, y] = x$ in $Z_P^*$ such that $b^x \bmod P = y$.

*The discrete logarithm assumption.* (This asserts that any procedure for solving the discrete logarithm problem will be inefficient for a fraction of the inputs.) Let $\mathbf{P}[b, P, y]$ be a (probabilistic) procedure for solving the discrete logarithm problem. Let $0 < \delta < 1$ be a fixed constant, and let $t$ be a fixed positive integer. Let poly be a fixed polynomial. Then for all sufficiently large $n$, for all but $\delta$-fraction of $n$-bit primes $P$, for all primitive roots $b$ mod $P$, Probability {(expected) time to compute $\mathbf{P}[b, P, y] \geqq$ poly $(n) | y$ is selected uniformly from $Z_P^*\} > 1/n^t$.

2. *The quadratic residuacity problem* [Gauss]. Let $N$ be a product of two distinct odd primes. Exactly half the elements of $Z_N^*$ have Jacobi symbol +1, the other half have Jacobi symbol −1. Denote the former by $Z_N^*(+1)$ and the latter by $Z_N^*(-1)$. None of the elements of $Z_N^*(-1)$ and exactly half the elements of $Z_N^*(+1)$ are quadratic residues. The *quadratic residuacity problem* with parameters $N$ and $x$ consists in deciding, for $x$ in $Z_N^*(+1)$, whether or not $x$ is a quadratic residue.

*The quadratic residuacity assumption* (QRA). (This asserts that any efficient procedure for guessing quadratic residuacity will be incorrect for a fraction of the inputs.) Let poly $(\cdot)$ be a polynomial. Let $\mathbf{P}[N, x]$ be any (probabilistic) poly-time

procedure which, on inputs $N$, $x$, each of length $n$, outputs 0 or 1. Let $0<\delta<1$ be a fixed constant, and let $t$ be a positive integer. Then for $n$ sufficiently large and for all but $\delta$ fraction of numbers $N$ of length $n$, $N$ a product of two distinct equal length odd primes, the probability that $\mathbf{P}[N, x]$ is incorrect in guessing quadratic residuacity (i.e., $\mathbf{P}[N, x]=0$ if $x \in Z_N^*(+1)$ is a quadratic residue mod $N$; 1 if not), given that $x$ is chosen uniformly from $Z_N^*(+1)$ and given the sequence of coin flips (in the case the procedure is probabilistic), exceeds $1/n^t$ in the sense that

$$\left(\frac{\sum_{x \in Z_N^*(+1)} \text{Prob}\,(\mathbf{P}[N, x] \text{ is incorrect})}{\varphi(N)/2}\right) > 1/n^t.$$

By Lemma 3, the "$1/n^t$" is replaceable by "$(1/2)-(1/n^t)$."

**6. The $1/P$ generator is predictable.** Let $P$ and $b$ be relatively prime integers $>1$ and $r_0$ an integer in the range $0<r_0<P$. Denote the expansion of $r_0/P$ to base $b$ by

$$r_0/P = .q_1 q_2 q_3 \cdots \tag{1}$$

where $0 \leqq q_i < b$. Since $b$ is prime to $P$, the expansion is periodic. Then, for $m \geqq 0$,

$$(b^m \cdot r_0)/P = q_1 \cdots q_m . q_{m+1} q_{m+2} \cdots = (q_1 \cdots q_m) + r_m/P \tag{2}$$

where

$$0 < r_m = b^m r_0 \bmod P < P \tag{3}$$

and

$$0 < r_m/P = .q_{m+1} q_{m+2} \cdots = (b^m \cdot r_0/P) \bmod 1 < 1. \tag{4}$$

Here, $q_1, q_2, \cdots$ are (quotient) *digits* base $b$ and $q_1 q_2 \cdots$ denotes their concatenation, whereas $r_m$, the $m$th remainder (of $r_0/P$ base $b$), is an *integer* whose length (base $b$) is less than or equal to the length of $P$: $|r_m|_b \leqq |P|$, where in this section $|P|$ denotes $|P|_b$. Recall for $r_0 \in Z_P^*$, (1) defines the pseudo-random sequence generated by the $1/P$ generator with input $r_0/P$.

There are several reasons one might consider the $1/P$ generator a good pseudo-random sequence generator: if the parameter $P$ is a prime, and $b$ is a primitive root mod $P$, the sequences produced have long periods and nice distribution properties (Theorem 1 below)[1]. In addition, these sequences possess certain hard-to-infer properties. For example, given a remainder $r$ generated during the expansion of $1/P$ base $b$, it is hard, in general, to find any index $m$ such that $r_m = r$. This is because $r_m = b^m \bmod P$, so $m$ is the discrete logarithm of $r$ mod $P$. It follows (Theorem 2, problem 1) that, given a string of quotient digits $q_{m+1} q_{m+2} \cdots q_{m+k}$ ($k \leqq \text{poly}\,(|P|)$), it is hard in general to find its location in the sequence.

---

[1] We remark that it would be natural to restrict the $1/P$ generator (base $b$) to the seed space $Y = \{(P, r) | P$ is an odd prime, $b$ is a primitive root mod $P$, $r \in Z_P^*\}$ with the product distribution: for each $(P, r) \in Y^n$, let $\mu_n(P, r) = u_n(P) \cdot v_P(r)$, where $u_n$ is the uniform distribution on the parameters of length $n$ and $v_P$ is uniform on $Z_P^*$. Then, on reasonable conjecture, $\{\mu_n\}_{n \in Z^+}$ is accessible on $Y$ since: a) E. Artin's conjecture and the prime number theorem imply that if $b$ is not a square, then the cardinality of $\{P | P$ is a prime of length $n$ and $b$ is a primitive root mod $P\}$ is more than $(1/3) \cdot (2^n/n)$, asymptotically as $n$ goes to infinity [Shanks p. 81]. And, there are (Monte-Carlo) probabilistic polynomial-time procedures for b) testing primality [Strassen–Solovay]; c) testing if $b$ is a primitive root mod $P$, given $P$ and the factorization of $P-1$ [LeVeque, Thm. 4.8]; d) producing, for any $k$, $k$ bit integers in factored form according to the uniform probability distribution [Bach]; and e) computing greatest common divisors.

On the other hand, Theorem 2 will give a sense, which is correct, that the $1/P$ generator yields a poor pseudo-random sequence: from knowledge of $P$ and any $|P|$-long segment of the expansion of $r_0/P$ base $b$, one can efficiently extend the segment backwards and forwards (problem 2). More surprisingly (problem 4), from knowledge of any $2|P|+1$ successive elements of the sequence, but *not* $P$, one can efficiently reconstruct $P$, and hence efficiently continue the sequence in either direction.

It follows that there is a simple efficient statistical test for deciding whether a $3n$-long string of digits has either been generated by the expansion of $1/P$ base $b$, for some prime $P$ of length $n$, or has been generated at random (uniform probability distribution), given that it was produced in one of those two ways. Use $2n+1$ of the given $3n$ digits to recover the suspected $P$; use this $P$ to generate $3n$ digits; then compare the generated digits with the $3n$ given digits: if they agree, the string has probably (with probability $\geqq 1-1/2^{n-1}$) been generated using the $1/P$ generator.

To lead up to Theorem 1, we consider the following types of sequences (closely related to maximum-length shift register sequences [Golomb]).

DEFINITION. Let $P$, $b$ denote arbitrary positive integers. A (*generalized*) *de Bruijn sequence of period* $P-1$, *base* $b$, is a sequence $q_1q_2\cdots$ of $b$-ary digits (i.e., $0\leqq q_i<b$ for all $i$) of period $P-1$ such that (1) every $b$-ary string of length $|P|-1$ occurs at least once in the sequence, and (2) every $b$-ary string of length $|P|$ occurs at most once in any given period of the sequence.

THEOREM 1. *Let* $P=$ *prime. Let* $b\in\{1,2,\cdots,P-1\}$ *be a primitive root* mod $P$. *Let* $r_0\in\{1,2,\cdots,P-1\}$. *Then the pseudo-random sequence generated by the* $1/P$ *generator* (*base* $b$) *with input* $r_0/P$ *is a* (*generalized*) *de Bruijn sequence of period* $P-1$, *base* $b$.

*Proof.* Since $r_m=b^m\cdot r_0 \bmod P$ and $b$ is a primitive root mod $P$, the sequence of remainders $r_m$ (generated during the expansion of $1/P$ base $b$) is periodic with period $P-1$, the remainders in any period are distinct, and $\{r_m|1\leqq m\leqq P-1\}=\{1,2,\cdots,P-1\}$.

Similarly, the sequence of quotients $r_m/P$ is periodic with period $P-1$, the quotients in any period are distinct, and

$$\{r_m/P|1\leqq m\leqq P-1\}=\{1/P,2/P,\cdots,(P-1)/P\}. \tag{5}$$

Therefore, the sequence of quotient digits $q_m$ is periodic with period at most $P-1$. If the period were less than $P-1$, then there would be integers $0\leqq m_1<m_2<P-1$ such that $.q_{m_1+1}q_{m_1+2}\cdots=.q_{m_2+1}q_{m_2+2}\cdots$. Since $r_m/P=.q_{m+1}q_{m+2}\cdots$, we would have $r_{m_1}/P=r_{m_2}/P$, a contradiction. Therefore the period is $P-1$. [Gauss]

Now, a string $a_1\cdots a_s$ of $s$ $b$-ary digits appears somewhere in the expansion of $r_0/P$ if and only if it appears as an initial string in the expansion of $r_m/P$ for some $1\leqq m\leqq P-1$ if and only if (by (5)) it appears as an initial string in the expansion of $k/P$ for some $1\leqq k\leqq P-1$. But also, the set of $b$-ary strings of length $s$ correspond exactly to the subintervals of the unit interval $[0,1)$ of the form $[l/b^s,(l+1)/b^s)$ where $l$ is an integer, $0\leqq l<b^s$. Since $1/P<1/b^{|P|-1}$, there is for each $l$, at least one $k$, $1\leqq k\leqq P-1$ such that $k/P\in[l/b^{|P|-1},(l+1)/b^{|P|-1})$ and so we have property 1. Since $1/b^{|P|}<1/P$, there is for each $l$ at most one $k$, $1\leqq k\leqq P-1$ such that $k/P\in[l/b^{|P|},(l+1)/b^{|P|})$, and so we get property 2. QED

So, if $P$ is prime and $b$ is a primitive root mod $P$, it follows from Theorem 1 concerning de Bruijn property 1 (and Artin's conjecture—see footnote 2 concerning that conjecture) that neither $|P|-1$ successive digits of quotient, $q_{m+1}\cdots q_{m+|P|-1}$, nor (the approximately $|P|-1$ successive digits of) a remainder, $r_m$, are enough to construct $P$, or to extend the sequence, on purely information-theoretic grounds. In contrast, it

will follow from Theorem 2 below that (various combinations of) approximately $2|P|$ digits of information are sufficient to efficiently extend the sequence in either direction.

THEOREM 2. *Let $P$ and $b$ be relatively prime integers $>1$ ($P$ not necessarily prime), and let $r_0$ be an integer in the range $0<r_0<P$. The following problems are solvable in polynomial($|P|$)-time.*

*Problem* 1. *Choose a polynomial,* poly $(\cdot)$, *and hold it fixed.*
INPUT: $P$, $b$, *remainder* $r_m$, *positive integer* $k \leqq \text{poly}(|P|)$.
OUTPUT: $r_{m-1}$, $r_{m+k}$; $q_m q_{m+1} \cdots q_{m+k}$.

*Problem* 2 [Gauss]. *This is a computational version of Theorem* 1 *concerning de Bruijn property* 2. (*A similar algorithm gives the computational version of property* 1.)
INPUT: $P$, $b$, $|P|$ *successive digits of quotient* $q_{m+1}q_{m+2}\cdots q_{m+|P|}$.
OUTPUT: $r_m$ (*and hence, by problem* 1, $r_{m+|P|}$ and $q_m$, $q_{m+|P|+1}$).

*Problem* 3. *We assume that $P$ is relatively prime to each of* $1, 2, \cdots, b$ (*to ensure that the output is the unique $P$ that generated $r_m$ and $r_{m+1}$*).
INPUT: $b$, $r_m$, $r_{m+1}$ *such that* $r_m \cdot b \neq r_{m+1}$ (i.e. $r_m \geqq P/b$).
OUTPUT: $P$ (*and therefore also, by problem* 1, $q_m q_{m+1}\cdots q_{m+|P|}$).

*Problem* 4. *We assume that $r_0$ is relatively prime to $P$* (e.g., $r_0 = 1$).
INPUT: $b$; $k$ *quotient digits,* $q_{m+1}q_{m+2}\cdots q_{m+k}$, *where* $k = \lceil \log_b (2P^2) \rceil$ *and $m$ is arbitrary.* (*Note that* $k \leqq 2|P|+1$).
OUTPUT: $P$; $r_m$ (*and hence by problem* 1, $q_m$ *and* $q_{m+k+1}$).

*Proof.* To solve problem 1: $r_{m+k} = b^k r_m \bmod P$ and $r_{m-1} = b^{-1} r_m \bmod P$ where $b^{-1}$ is the inverse of $b$ mod $P$. We note that

$$(b^k r_m)/P = q_{m+1}\cdots q_{m+k} + r_{m+k}/P. \tag{6}$$

So, $q_m \cdots q_{m+k} = \lfloor (b^{k+1} r_{m-1})/P \rfloor$. (By convention, we do not drop initial digits in a concatenation of quotient digits, e.g., in (6).)

To solve problem 2: By (6), $r_m = (q_{m+1}\cdots q_{m+|P|})\cdot P/b^{|P|} + (r_{m+|P|})/b^{|P|}$. Since $r_{m+|P|} < P < b^{|P|}$, $r_m = \lceil (q_{m+1}\cdots q_{m+|P|})\cdot P/b^{|P|} \rceil$.

In problems 3 and 4, the number $P$ is not available and must be constructed.

To solve problem 3: By (6) with $k=1$, $b\cdot r_m - r_{m+1} = q_{m+1}\cdot P$ where $0 \leqq q_{m+1} < b$. Actually, $0 < q_{m+1}$, since, by assumption, $b\cdot r_m \neq r_{m+1}$. Therefore, $P$ equals some integer in the sequence of real numbers $(b\cdot r_m - r_{m+1})/1$, $(b\cdot r_m - r_{m+1})/2, \cdots, (b\cdot r_m - r_{m+1})/b - 1$. Select any integer $P$ in the sequence such that $P$ is relatively prime to $1, 2, \cdots, b$. Such an integer $P$ is unique; for suppose to the contrary that $P$, $Q$ are two such integers relatively prime to each of $1, 2, \cdots, b$. Then $P\cdot(i) = Q\cdot(j)$ for some $0 < i, j < b$. Without loss of generality, suppose $P < Q$. $Q$ is relatively prime to each of $1, 2, \cdots, b$, so $\gcd(Q, i) = 1$, so $Q|P$, so $Q \leqq P$, which is a contradiction.

The solution to problem 4, which is very pretty, is by continued fractions: By (6), $r_m/P = q_{m+1}\cdots q_{m+k}/b^k + \varepsilon$ where $0 \leqq \varepsilon < 1/b^k$. By [LeVeque, p. 237, Thm. 9.10], the continued fraction expansion of $q_{m+1}\cdots q_{m+k}/b^k$ has convergent $r_m/P$ if $1/b^k \leqq 1/2P^2$, i.e., $2P^2 \leqq b^k$, i.e., $\log_b (2P^2) \leqq k$, as postulated. So $r_m/P = A_i/B_i$ for one of the convergents $A_1/B_1, A_2/B_2, \cdots$ of the fraction $q_{m+1}\cdots q_{m+k}/b^k$. Since both $b$ and $r_0$ are relatively prime to $P$, it follows (from (3)) that $\gcd(r_m, P) = 1$, so $r_m = A_i$ and $P = B_i$.

It remains to show that $r_m$ and $P$ can be obtained by generating the above convergents until for some $i$ the first $k$ digits of $A_i/B_i$ are $q_{m+1}\cdots q_{m+k}$, at which point $r_m = A_i$ and $P = B_i$. To see why, recall that the continued fraction $q_{m+1}\cdots q_{m+k}/b^k = 1/a_1 + 1/a_2 + 1/a_3 + \cdots 1/a_i + \cdots$ has convergents $A_1/B_1 = 1/a_1$, $A_2/B_2 = a_2/(a_1a_2+1), \cdots, A_i/B_i = (a_iA_{i-1} + A_{i-2})/(a_iB_{i-1} + B_{i-2}), \cdots$. Here, the $B_i$

are strictly increasing with $i$. Since for some $i$, $A_i/B_i = r_m/P$, this procedure for obtaining $r_m$ and $P$ will never go beyond $A_i/B_i = r_m/P$. To see that the procedure generates convergents to the point where $A_i/B_i = r_m/P$, note that when $A_j/B_j = q_{m+1} \cdots q_{m+k} \cdots$, the error is sufficiently small to ensure that $A_j/B_j = r_m/P$.

Since $A_i$ and $B_i$ grow exponentially, $P = B_i$ and $r_m = A_i$ can be computed in polynomial($|B_i|$), in particular in $O$(number of steps to compute the $i$th Fibonacci number), and therefore in polynomial($|P|$) steps. This solves problem 4. QED

*Example.* Let $b = 10$ and $P = 503$. Then $P$ is a prime and $b$ is a primitive root mod $P$, so the $1/P$ *generator* with input $1/503$ quickly generates a sequence of base 10 digits with period 502. This sequence is

00198 80715 70576 54075 54671 96819 08548 70775 34791 25248 50894 63220 67594 **43339 9**6023
85685 88469 18489 06560 63618 29025 84493 04174 95029 82107 35586 48111 33200 79522 86282
30616 30218 68787 27634 19483 10139 16500 99403 57852 88270 37773 35984 09542 74353 87673
95626 24254 47316 10337 97216 69980 11928 42942 34592 44532 80318 09145 12922 46520 87475
14910 53677 93240 55666 00397 61431 41153 08151 09343 93638 17097 41550 69582 50497 01789
26441 35188 86679 92047 71371 76938 36978 13121 27236 58051 68986 08349 90059 64214 71172
96222 66401 59045 72564 61232 60437 37574 55268 38966 20278 33001 98807 $\cdots$

Since $|503| = 3$, every string of two decimal digits occurs at least once in the above sequence, and every string of three decimal digits occurs at most once in any period of the sequence.

Since $k = \lceil \log_{10}(2 \cdot 503^2) \rceil = 6$, we can, from any segment of length 6 of the above sequence, efficiently recover $P$, and then quickly extend the segment in either direction. For example, consider the segment 433399 (shown in bold type above). The continued fraction expansion of .433399 is $433{,}399/1{,}000{,}000 = 1/2+1/3+1/3+1/1+1/16+1/6+1/1+1/1+1/358+\cdots$, and its first five convergents are: $\frac{1}{2} = .5$; $\frac{3}{7} = .48\cdots$; $\frac{10}{23} = .434\cdots$; $\frac{13}{30} = .43333\cdots$; $\frac{218}{503} = .4333996\cdots$. At last, the first 6 digits agree with the segment 433399. So we get $P = 503$ and $r_m = 218$ (and so $r_{m-1} = 10^{-1} \cdot r_m \bmod 503 = 151 \cdot 218 \bmod 503 = 223$). In this way, we can extend the given segment, 433399, forwards and backwards.

**7. The $x^2 \bmod N$ generator is unpredictable.** In this section we elaborate on properties of the $x^2 \bmod N$ pseudo-random sequence generator, and prove (modulo the QRA) that it is polynomial-time unpredictable (Theorem 4, this section).

First we recall some number-theoretic facts. Suppose $N = P \cdot Q$ where $P$ and $Q$ are distinct odd primes. Let $Z_N^* = \{\text{integers } x | 0 < x < N \text{ and } \gcd(x, N) = 1\}$. Then $QR_N$, the set of quadratic residues mod $N$, form a multiplicative subgroup of $Z_N^*$ of order $\varphi(N)/4$ (where $\varphi(N)$ is the cardinality of $Z_N^*$). Each quadratic residue $x^2 \bmod N$ has four distinct square roots, $\pm x \bmod N$, $\pm y \bmod N$. If we also assume, as we shall for the rest of this paper, that $P \equiv Q \equiv 3 \bmod 4$, then each quadratic residue mod $N$ has *exactly* one square root which is also a quadratic residue (see Lemma 1, this section). In other words, squaring mod $N$ is a $1-1$ map of $QR_N$ onto $QR_N$. (Comment: half the primes of length $n$ are congruent to 3 mod 4 asymptotically as $n \to \infty$ [LeVeque], so there are plenty such $N$.)

We now investigate what properties can be inferred about sequences produced by the $x^2 \bmod N$ generator, given varying amounts of information. In the following, $N$ is of the *prescribed form*, that is to say, $N = P \cdot Q$ where $P$, $Q$ are distinct primes both congruent to 3 mod 4. Also, $x_i$ is a quadratic residue mod $N$, $x_{i+1} = x_i^2 \bmod N$ and $b_i = \text{parity}(x_i)$:

1. Clearly, knowledge of $N$ is sufficient to efficiently generate sequences $x_0, x_1, x_2, \cdots$ (and hence sequences $b_0 b_1 b_2 \cdots$) in the forward direction, starting from any given seed $x_0$. The number of steps per output is $O(|N|^{1+\varepsilon})$ using fast multiplication.

2. Given $N$, the factors of $N$ are necessary and sufficient to efficiently generate the $x^2 \bmod N$ sequences in the reverse direction, $x_0, x_{-1}, x_{-2}, \cdots$, starting from any given seed $x_0$. (See proof below).

3. What is more, the factors of $N$ are necessary—assuming they are necessary for deciding quadratic residuacity of an $x$ in $Z_N^*(+1)$—to have even an $\varepsilon$-advantage in guessing in polynomial time the parity of $x_{-1}$, given $N$ and given $x_0$ chosen "at random" from $QR_N$. (Note that to choose a quadratic residue at random with the uniform probability distribution from $QR_N$, it is sufficient to choose $x$ at random (with the uniform probability distribution) from $Z_N^*$ and square it mod $N$).[2]

To see Claim 2 above, we first prove the following:

LEMMA 1. *If $N = P \cdot Q$ where $P$ and $Q$ are distinct primes such that $P \equiv Q \equiv 3 \bmod 4$, then each quadratic residue* mod $N$ *has exactly one square root that is a quadratic residue.*

*Proof.* Whenever $N$ is a product of two distinct odd primes, every quadratic residue mod $N$ has four square roots, $\pm x$ and $\pm y$. Since $N \equiv 1 \bmod 4$, their Jacobi symbols satisfy $(+x/N) = (-x/N)$ and $(+y/N) = (-y/N)$. Since $P \equiv 3 \bmod 4$, $(+x/N) \neq (+y/N)$ (this can easily be proved from the fact that $\gcd(x+y, N) = P$ and $\gcd(x-y, N) = Q$, whence $x+y = kP$ and $x-y = lQ$, whence $(x/P) = (-y/P)$ and $(x/Q) = (y/Q)$). Thus $(+x/N) = (-x/N) \neq (+y/N) = (-y/N)$. Eliminating the two roots, say $\pm y$, with Jacobi symbol $-1$ with respect to $N$, we are left with the two roots $\pm x$ having Jacobi symbol $+1$ with respect to $N$. Exactly one of these roots has Jacobi symbol $+1$ with respect to both $P$ and $Q$, because $P \equiv 3 \bmod 4$, and this one and this one only is a quadratic residue mod $N$. QED

The necessity (of knowing the factors of $N$) now follows: Suppose we can efficiently generate such sequences in the reverse direction. To factor $N$, select an $x$ in $Z_N^*$ whose Jacobi symbol is $(x/N) = -1$. Let $x_0 = x^2 \bmod N$ and compute $x_{-1}$. Then efficiently compute $\gcd(x + x_{-1}, N) = P$ or $Q$. We can sharpen this argument to show [Rabin '79] that the ability to compute $x_{-1}$ for even a fraction of seeds $x_0$ will enable us to factor $N$ efficiently with high probability.

On the other hand, if we know the factors of $N$ we can use the algorithm described in Theorem 3 (below) to efficiently generate sequences backwards:

THEOREM 3. *There is an efficient deterministic algorithm $A$ which when given $N$ (of the prescribed form), the prime factors of $N$ and any quadratic residue $x_0$ in $Z_N^*$, efficiently computes the unique quadratic residue $x_{-1}$* mod $N$ *such that $(x_{-1})^2 \bmod N = x_0$. Thus,*

$$A(P, Q, x_0) = x_{-1}.$$

*Proof.* By Lemma 1, the map from the quadratic residues mod $N$ into the quadratic residues mod $N$, $f: x \to x^2 \bmod N$, is $1-1$ onto. The algorithm $A$ can now be described as follows:

INPUT: $P, Q$ = two distinct primes congruent to 3 mod 4; $x_0$ = a quadratic residue mod $N$, where $N = P \cdot Q$.

OUTPUT: A quadratic residue $x_{-1}$ mod $N$ whose square mod $N$ is $x_0$.

Compute $x_P = \sqrt{x_0} \bmod P$ such that $(x_P/P) = +1$, where $\sqrt{x_0} \bmod P$ denotes an integer in $Z_P^*$ whose square mod $P$ is $x_0$:

$\sqrt{x_0} \bmod P = \pm x_0^{(P+1)/4} \bmod P$ (for $P \equiv 3 \bmod 4$). Compute $x_Q = \sqrt{x_0} \bmod Q$. Use the Euclidean algorithm to construct integers $u, v$ such that $P \cdot u + Q \cdot v = 1$, and from

[2] A more formal statement of claim 3: Modulo the QRA, given a polynomial poly, a constant $0 < \delta < 1$, and a positive integer $t$, if $\mathbf{P}[N, x_0]$ is a probabilistic poly-time procedure for guessing the parity of $x_{-1}$ given $x_0$ in $QR_N$, then $(\sum_{x_0 \in QR_N} \text{Prob}[\mathbf{P}[N, x_0] = \text{Parity}(x_{-1})])/(\varphi(N)/4) < (1/2) + (1/n^t)$ for sufficiently large $n$, and all but $\delta$ fraction of prescribed integers $N$ of length $n$.

that obtain the particular number, $x_N = \pm x_P \cdot Q \cdot v \pm x_Q \cdot P \cdot u = \sqrt{x_0} \bmod N$, that is a square root of $x_0 \bmod N$, and that is also a quadratic residue with respect to both $P$ and $Q$ and therefore with respect to $N$. QED

To see Claim 3 above, we start with the following definition.

DEFINITION. Given a polynomial poly and $0 < \varepsilon \leqq 1/2$, a $0-1$ valued probabilistic poly-time procedure $\mathbf{P}(\cdot, \cdot)$ has an *$\varepsilon$-advantage for $N$ in guessing parity* (of $x_{-1}$ given arbitrary $x_0$ in $QR_N$) if and only if $(\sum_{x_0 \in QR_N} \text{Prob}[\mathbf{P}[N, x_0] = \text{Parity}(x_{-1})])/(\varphi(N)/4) \geqq (1/2) + \varepsilon$. In a similar fashion, we can define a procedure having an *$\varepsilon$-advantage for $N$ in guessing quadratic residuacity* (of arbitrary $x \in Z_N^*(+1)$) [Goldwasser–Micali]. The $1/2 + \varepsilon$ makes sense in the second definition since exactly half the elements in $Z_N^*(+1)$ are quadratic residues.

LEMMA 2. *An $\varepsilon$-advantage for guessing parity (of $x_{-1}$ given quadratic residue $x_0$) can be converted, efficiently and uniformly, to an $\varepsilon$-advantage for guessing quadratic residuacity (of $x$ in $Z_N^*(+1)$).*[3]

*Proof.* Let $x \in Z_N^*(+1)$ be an element whose quadratic residuacity mod $N$ is to be determined. Set $x_0 = x^2 \bmod N$. Since $P \equiv Q \equiv 3 \bmod 4$, the square roots of $x^2 \bmod N$ that are in $Z_N^*(+1)$ are $x$ and $N - x$ (see proof of Lemma 1), and since $N$ is odd, each of these square roots has opposite parity. Only one of these square roots is a quadratic residue (i.e., equal to $x_{-1}$), and only one of these has parity equal to parity $(x_{-1})$. Therefore, $x$ is a quadratic residue mod $N$ if and only if $x = x_{-1}$ if and only if parity $(x)$ = parity $(x_{-1})$. QED

LEMMA 3 (Goldwasser and Micali). *An $\varepsilon$-advantage for guessing quadratic residuacity can be amplified to a $1/2 - \varepsilon$ advantage, uniformly and efficiently.*[4]

*Idea of proof.* Suppose $\mathbf{P}[N, x]$ is a probabilistic poly-time procedure that has an $\varepsilon$-advantage for $N$ in guessing quadratic residuacity. Then we can in polynomial time sample (uniformly) the elements $x$ of $QR_N$ and of $Z_N^*(+1) - QR_N$ (by selecting a number at random from $Z_N^*$, squaring it, then taking its negative mod $N$), and estimate constants $A$ and $B$ such that

$$\left(\frac{\sum_{x \in QR_N} \text{Prob}[\mathbf{P}[N, x] = 1]}{\varphi(N)/4}\right) \approx A \quad \text{and} \quad \left(\frac{\sum_{x \in Z_N^*(+1) - QR_N} \text{Prob}[\mathbf{P}[N, x] = 1]}{\varphi(N)/4}\right) \approx B,$$

and $A\text{-}B \geqq 2\varepsilon$ (approximately). Now let $x \in Z_N^*(+1)$ be an element whose quadratic residuacity mod $N$ is to be determined. To this end, select $r$'s independently and at random with uniform probability from $Z_N^*$. Compute $x \cdot r^2 \bmod N$. [Comment: For $x \in QR_N$, $x \cdot r^2 \bmod N$ is uniformly distributed over $QR_N$; for $x \notin QR_N$, $x \cdot r^2 \bmod N$ is uniformly distributed over $Z_N^*(+1) - QR_N$.] Test each of the resulting numbers, $x \cdot r^2 \bmod N$, for quadratic residuacity by checking if $\mathbf{P}[N, xr^2] = 1$ (using sequences of coin tosses as may be required by $\mathbf{P}$). Compare the resulting fraction of favorable outcomes to the fractions $A$ and $B$ in order to get an amplified advantage in guessing quadratic residuacity of $x$. QED

---

[3] A more formal statement of Lemma 2: Given poly, $0 < \varepsilon(n) \leqq 1/2$, $\mathbf{N}$ = a set of integers $N$ of the prescribed type. If there is a probabilistic poly-time procedure that has an $\varepsilon(|N|)$-advantage for each $N \in \mathbf{N}$ in guessing parity (of $x_{-1}$ given an arbitrary $x_0 \in QR_N$), then there is a polynomial poly′ and a probabilistic poly′ procedure that has an $\varepsilon(|N|)$-advantage for $N \in \mathbf{N}$ in guessing quadratic residuacity (of arbitrary $x$ in $Z_N^*(+1)$).

[4] A more formal statement of Lemma 3: Given poly, $t$ = a positive integer, and $\mathbf{N}$ = a set of integers $N$ of the prescribed type. If there is a probabilistic poly-time procedure that has a $1/|N|^t$ advantage for $N \in \mathbf{N}$ in guessing quadratic residuacity (of $x$ in $Z_N^*(+1)$, then for any positive integer $t'$ there is a polynomial poly′ and a probabilistic poly′-time procedure that has a $1/2 - 1/|N|^{t'}$ advantage for $N \in \mathbf{N}$ in guessing quadratic residuacity (of $x \in Z_N^*(+1)$).

CLAIM 3 follows: Suppose to the contrary that **P** is a probabilistic poly procedure that has a $1/n^t$ advantage in determining parity (for infinitely many $n$, and for more than $\delta$ of prescribed numbers $N$ of length $n$). Then convert **P** (Lemma 2) to a probabilistic poly′ procedure **P**′ for determining quadratic residuacity that has an amplified advantage (Lemma 3) of $1/2-1/n^{t'}$ (for these same integers $N$). This contradicts the quadratic residuacity assumption.

Leading up to Theorem 4 we make the following definition.

DEFINITION.

1. A *predictor* $\mathbf{P}(\cdot,\cdot)$ is a probabilistic poly-time procedure that on inputs $N$, $b_1 \cdots b_k$, with $b_i \in \{0,1\}$ and $k \leqq \text{poly}(|N|)$, outputs a 0 or 1 (the output may depend on the sequence of coin-tosses that the probabilistic algorithm makes).

2. **P** has an *$\varepsilon$-advantage* for $N$ in predicting to the left sequences produced by the $x^2 \bmod N$ generator if and only if for some $k \leqq \text{poly}(|N|)$, $\sum_{x \in QR_N} \text{Prob}[P(N, b_1(x), \cdots b_k(x)) = b_0(x)]/(\varphi(N)/4) \geqq (1/2)+\varepsilon$, where $b_i(x) = \text{parity}(x^{2^i} \bmod N)$.

THEOREM 4. *Modulo the* QRA, *the $x^2 \bmod N$ generator is an unpredictable (cryptographically secure) pseudo-random sequence generator. That is to say, for each probabilistic poly-time predictor* **P**, *each constant $0<\delta<1$, and positive integer $t$,* **P** *has at most a $1/n^t$ advantage for $N$ in predicting sequences to the left (for sufficiently large $n$ and for all but a $\delta$ fraction of prescribed numbers $N$ of length $n$).*

*Proof.* Suppose we have a predictor for the $x^2 \bmod N$ generator with an $\varepsilon$-advantage for $N$. This can be converted efficiently and uniformly into a procedure with an $\varepsilon$-advantage in guessing parity (of $x_{-1}$ given arbitrary $x_0$ in $QR_N$). To see this, suppose we are given $x_0 \in QR_N$. From seed $x_0$ generate the sequences $b_0 b_1 b_2 \cdots$. Then parity $(x_{-1}) = b_{-1}$.

Now convert (Lemma 2) to a procedure for guessing quadratic residuacity with an amplified advantage (Lemma 3) to get a contradiction to the quadratic residuacity assumption. QED

It follows from a fundamental theorem of Yao [Yao] that, under the QRA, the sequences produced by the $x^2 \bmod N$ generator pass every probabilistic polynomial-time statistical test (roughly speaking, these sequences cannot be distinguished by any poly $(n)$-time statistical test—with more than a negligible advantage—from sequences produced by successive flips of a fair coin). More precisely, what does this mean? Yao gives a very general definition of the concept of a probabilistic poly-time statistical test, but the following definition adequately describes such a test for our purpose: formally, a *probabilistic poly-time statistical test*, $T$, is a probabilistic poly-time algorithm that assigns to every input string in $\{0,1\}^*$ a real number in the unit interval $[0,1]$ (the particular value depends in general on the sequence of coin flips made by the algorithm). Let $\alpha_m$ denote the average value that such $T$ assigns to a random $m$-bit string (chosen with uniform probability from among all $m$-bit strings). We say that a *pseudo-random sequence generator passes test $T$* if, for every positive integer $t$, the average value, over all seeds of length $n$, that the statistical test assigns to a poly $(n)$-bit pseudo-random string (produced by the given generator) lies in the interval $\alpha_{\text{poly}(n)} \pm 1/n^t$ for all sufficiently large $n$. If a generator does not pass test $T$, then we say that $T$ has an *advantage* in distinguishing between the pseudo-random bits produced by the generator and truly random sequences of bits.

THEOREM 5 (following Yao). *Modulo the* QRA, *the sequences produced by the $x^2 \bmod N$ generator pass every probabilistic polynomial time statistical test.*

*Idea of proof.* Here and in the sequel, we use $\text{poly}_1$, $\text{poly}_2$, $\cdots$ to denote distinct polynomials. Suppose there were a probabilistic $\text{poly}_1$-time test $T$ that, for infinitely many $n$, has an advantage in distinguishing between the pseudo-random sequences of

length $\text{poly}_1(n)$ produced (from random seeds of length $n$) by the $x^2 \bmod N$ generator and truly random sequences of bits of the same $\text{poly}_1(n)$ length. Then for some positive integer $t$ and infinitely many $n$, the average value that $T$ assigns to the pseudo-random sequences of length $\text{poly}_1(n)$ (generated from seeds of length $n$) lies outside, say above, $\alpha_{\text{poly}_1(n)} \pm 1/n^t$, whereas the average value it assigns (truly) random sequences lies inside $\alpha_{\text{poly}_1(n)} \pm 1/n^{t+1}$. For each of these $n$, we can find integers $j, k \geqq 0, j+k = \text{poly}_1(n)$ (in probabilistic $\text{poly}_2(n)$-time) such that "with high probability" the average value that $T$ assigns to sequences in $A = \{r_1 \cdots r_j r_{j+1} b_1 \cdots b_k\}$ is closer to $\alpha_{\text{poly}_1(n)}$ by at least $1/(n^{t+1} \cdot \text{poly}_1(n))$ than the average higher value it assigns to sequences in $B = \{r_1 \cdots r_j b_0 b_1 \cdots b_k\}$—where the $b_0 \cdots b_k$ are sequences produced by the generator, the seed $x_0$ having been chosen uniformly at random, and the $r_1 \cdots r_{j+1}$ are sequences of independent random bits. Integers $j$, $k$ are found by trying different values of $j$, $k$, in each case sampling elements of the associated sets $A$, $B$, and applying $T$ to these samples. The Weak Law of Large Numbers assures that this can be done in probabilistic $\text{poly}_2(n)$-time.

We can convert $T$ into a predictor for the generator: Given a sequence $b_1 \cdots b_k$ produced by the generator, we submit a sample of $\text{poly}_3(n)$-many sequences of the form $r_1 \cdots r_j 0 b_1 \cdots b_k$ (where the $r_1, \cdots, r_j$ are independently chosen random bits) to test $T$ and estimate the average value, call it $\alpha^0$, assigned by $T$ to the entire set of these sequences. Then submit the corresponding sequences $r_1 \cdots r_j 1 b_1 \cdots b_k$ to $T$ and estimate the associated $\alpha^1$. $T$'s "advantage" in distinguishing between pseudo-random and random sequences can now be converted into an advantage in predicting $b_0$ correctly: Use a biased coin to predict $b_0$. Set the bias so that the coin has probability $\frac{1}{2}+\frac{1}{2}(\alpha^0-\alpha^1)$ of coming up heads. Toss the coin and if it comes up heads, predict $b_0 = 0$, else $b_0 = 1$. (It is tempting but incorrect to suggest that we predict $b_0 = 0$ if $\alpha^0$ is greater than $\alpha^1$, else $b_0 = 1$. Problems arise if for a *few* choices of $b_1 \cdots$, the random strings $r_1 \cdots r_j b_1 \cdots b_k$ give a correct *strong* bias toward $b_0 = 0$, whereas for *most* choices of $b_1' \cdots$, the strings $r_1 \cdots r_j b_1' \cdots b_k'$ give a *weak* bias toward $b_0' = 1$. One would end up giving wrong answers for the majority of strings $b_1' \cdots b_k'$. In this case, the expectation of predicting $b_0$ correctly would be less than $\frac{1}{2}$. The biased coin makes the expectation greater than $\frac{1}{2}$.) QED

*Remark.* We can construct another unpredictable generator as follows: recall that since $N \equiv 1 \bmod 4$, both $x$ and $-x$ (in $Z_N^*$) have the same Jacobi symbol, and since $N$ is odd, $x$ and $-x$ have opposite parity. Therefore, the parity property partitions $Z_N^*(+1)$ in half. In a similar fashion, the location property, where location $(x) = 0$ if $x < (N-1)/2$, 1 if $x \geqq (N-1)/2$, partitions $Z_N^*(+1)$ in half. Exactly one of $x$ and $-x$ is a quadratic residue; but which of the two is the quadratic residue is hard to decide. Thus we get the following.

THEOREM. *The modified $x^2 \bmod N$ generator, gotten by extracting the location bit at each stage (instead of parity) is cryptographically secure (modulo the quadratic residuacity assumption).*

*Conjecture.* The modified $x^2 \bmod N$ generator, gotten by extracting two bits at each stage, parity $(x)$ and location $(x)$, is cryptographically secure.

*Question.* Parity $(x)$ is the least significant bit of $x$; we can think of location $(x)$, in a sense, as the most significant bit. How many bits (and which ones) can we extract at each stage and still maintain cryptographic security?

**8. Lengths of periods (of the sequences produced by the $x^2 \bmod N$ generator).** What exactly is the period of the sequence generated by the $x^2 \bmod N$ generator? For quadratic residues $x_0 \bmod N$, let $\pi(x_0)$ be the period of the sequence $x_0, x_1, x_2, \cdots$, where $x_i = x_0^{2^i} \bmod N$. Since the $x^2 \bmod N$ generator is an unpredictable

pseudo-random sequence generator (modulo QRA), it follows that on the average, $\pi(x_0)$ will be long. In this section we investigate the precise lengths of these periods *without* relying on unproven assumptions (such as quadratic residuacity). To start, we show that the period is a divisor of $\lambda(\lambda(N))$.

DEFINITION. Let $M = 2^e * P_1^{e_1} * \cdots * P_k^{e_k}$, where $P_1, \cdots, P_k$ are distinct odd primes. Carmichael's $\lambda$-function is defined by

$$\lambda(2^e) = \begin{cases} 2^{e-1} & \text{if } e = 1 \text{ or } 2, \\ 2^{e-2} & \text{if } e > 2, \end{cases}$$

and $\lambda(M) = \text{lcm}\,[\lambda(2^e), (P_1-1) * P_1^{e_1-1}, \cdots, (P_k-1) * P_k^{e_k-1}]$. Carmichael [LeVeque], [Knuth] proves that $\lambda(M)$ is both the least common multiple and the supremum of the orders of the elements in $Z_M^*$. As corollary, Carmichael's extension of Euler's theorem asserts that $\alpha^{\lambda(M)} \equiv 1 \bmod M$ if $\gcd(\alpha, M) = 1$ [Knuth, vol. 2, p. 19].

The following theorem asserts that the period, $\pi(x_0)$, divides $\lambda(\lambda(N))$.

THEOREM 6. *Let $N$ be a number of the prescribed form (that is to say, $N = P * Q$ where $P$, $Q$ are distinct primes both congruent to* 3 mod 4). *Let $x_0$ be a quadratic residue* mod $N$. *Let $\pi = \pi(x_0) =$ period of the sequence $x_0, x_1, x_2, \cdots$. Then $\pi | \lambda(\lambda(N))$.*

*Proof.* Let $\text{ord}_N x$ denote the order of $x$ mod $N$. Then for $x \in Z_N^*(+1)$, $\text{ord}_N x$ is odd, because:

(1) $\text{ord}_N x_i = \text{ord}_N x_{i+1}$. This is because
  (i) $\text{ord}_N x_{i+1} | \text{ord}_N x_i$, and
  (ii) $x_0, x_1, \cdots$ cycles.

(2) for all positive integers $u$, $2^u \| \text{ord}_N x_i \Rightarrow 2^{u-1} \| \text{ord}_N x_{i+1}$. (Here, $2^u \| \text{ord}_N x$ means $2^u | \text{ord}_N x$ and $2^{u+1}$ does not divide $\text{ord}_N x$.)

Hence, by Carmichael's extension of Euler's theorem,

$$2^{\lambda(\text{ord}_N x_0)} \equiv 1 \bmod (\text{ord}_N x_0).$$

But $\pi$ is the least positive integer such that $2^\pi \equiv 1 \bmod (\text{ord}_N x_0)$, since $\pi$ is the least positive integer such that $x_0 = x_0^{2^\pi} \bmod N$.

Therefore, $\pi | \lambda(\text{ord}_N x_0)$. (This is a stronger result than the statement of the theorem!)

But $\lambda(\text{ord}_N x_0) | \lambda(\lambda(N))$ since $\text{ord}_N(x_0) | \lambda(N)$ for $x_0$ in $Z_N^*$.

Therefore, $\pi | \lambda(\lambda(N))$. QED

The following theorem provides conditions under which $\lambda(\lambda(N)) | \pi(x_0)$—and therefore $\lambda(\lambda(N)) = \pi(x_0)$.

THEOREM 7. *Let $N$ be a number of the prescribed form, $x_0$ a quadratic residue* mod $N$, *$\pi(x_0) =$ period of the sequence $x_0, x_1, \cdots$.*

1. *Choose $N$ so that* $\text{ord}_{\lambda(N)/2}(2) = \lambda(\lambda(N))$. (*Note: this equality frequently holds for prescribed $N$. See Theorem* 8.)

2. *Choose quadratic residue $x_0$ so that* $\text{ord}_N(x_0) = \lambda(N)/2$. (*Note: one can always choose a quadratic residue $x_0$ this way. See the paragraph below immediately following the proof of this theorem.*)

*Then* $\lambda(\lambda(N)) | \pi(x_0)$ (*and therefore* $\lambda(\lambda(N)) = \pi(x_0)$).

*Proof.* Recall that $x_i = (x_0)^{2^i} \bmod N$, and so $\pi =$ least positive integer such that $x_\pi = (x_0)^{2^\pi} \bmod N = x_0$.

Next note that $2^\pi \bmod (\lambda(N)/2) = 1$: By $2, \cdot \lambda(N)/2 =$ least positive integer such that $x_0^{\lambda(N)/2} \bmod N = 1$. But $x_0^{2^\pi} \bmod N = x_0$, so $x_0^{2^\pi - 1} \bmod N = 1$. Therefore, $\lambda(N)/2 | 2^\pi - 1$.

Finally, we show that $\lambda(\lambda(N))|\pi$: By 1, $\lambda(\lambda(N))$ = least positive integer such that $2^{\lambda(\lambda(N))} \bmod(\lambda(N)/2) = 1$, but (we just saw), $2^{\pi} \bmod(\lambda(N)/2) = 1$. Therefore $\lambda(\lambda(N))|\pi$. QED

Condition 2 of the above theorem holds for a substantial fraction of quadratic residues, $x_0$ in $Z_N^*$. Specifically, the number of quadratic residues in $Z_N^*$ that are of order $\lambda(N)/2 \bmod N$ is $\Omega(N/(\ln\ln N)^2)$ (where $f(n) = \Omega(g(n))$ means there exists a constant $c$ such that $f(n) > c \cdot g(n)$ for all sufficiently large $n$). To derive this lower bound, let $N = P \cdot Q$. Let $g_P$, $g_Q$ be generators mod $P$, $Q$ respectively. Let $a \equiv g_P \bmod P$, $\equiv g_Q \bmod Q$. It is easy to see that $\operatorname{ord}_N a = \operatorname{lcm}[P-1, Q-1] = \lambda(N)$. Now there are $\varphi(\varphi(P))$ generators mod $P$ and $\varphi(\varphi(Q))$ generators mod $Q$. By the Chinese remainder theorem, $Z_N^* = Z_P^* \times Z_Q^*$, so there are at least $\varphi(\varphi(P)) \cdot \varphi(\varphi(Q))$ elements in $Z_N^*$ of order $\lambda(N)$. But $\varphi(x) > x/(6 \ln\ln x)$ for all integers $x > 2$. Hence

$$\varphi(\varphi(P)) \cdot \varphi(\varphi(Q)) = \varphi(P-1) \cdot \varphi(Q-1) \geqq \frac{P-1}{6 \ln\ln(P-1)} \frac{Q-1}{6 \ln\ln(Q-1)}$$

$$\geqq \frac{N-P-Q+1}{[6 \ln\ln(N-1)]^2} = \Omega\left(\frac{N}{(\ln\ln N)^2}\right).$$

The map $x \to x^2 \bmod N$ is 4:1. Therefore, there are at least $\Omega(N/4(\ln\ln N)^2)$ quadratic residues in $Z_N^*$ of order $\lambda(N)/2$.

Condition 1 of the above theorem is harder to ensure in general. The following definition and theorem give conditions of special interest for our applications, under which condition 1 will hold.

DEFINITION. A prime $P$ is *special* if $P = 2P_1 + 1$ and $P_1 = 2P_2 + 1$ where $P_1$, $P_2$ are odd primes. A number $N = P * Q$ is a *special* number of the prescribed form if and only if $P$, $Q$ are distinct odd primes both congruent to 3 mod 4, and $P$, $Q$ are both special (note: distinctness of $P$ and $Q$ implies that $P_2 \neq Q_2$).

*Example.* The primes 2879, 1439, 719, 359, 179, 89, are special. The number $N = 23 * 47$ is a special number of the prescribed form.

*Remark.* It is reasonable to expect [Shanks] that the fraction of $n$-bit numbers that are special primes is asymptotically $1/((\ln P)(\ln P_1)(\ln P_2))$, which is asymptotically $1/(n^3 \ln^3 2)$ since $2^n < P < 2^{n+1}$, $2^{n-1} < P_1 < 2^n$, and $2^{n-2} < P_2 < 2^{n-1}$. It follows that there is an efficient, i.e., polynomial $(n)$, probabilistic algorithm to find special $n$-bit primes: simply generate $n$ bit numbers at random and use a probabilistic primality test [Strassen–Solovay], [Miller], [Rabin '80] to select the ones that are special.

THEOREM 8. *Suppose $N$ is a special number of the prescribed form, and that 2 is a quadratic residue with respect to at most one of $P_1$, $Q_1$.*[5] *Then* $\operatorname{ord}_{\lambda(N)/2}(2) = \lambda(\lambda(N))$ *(and therefore $\lambda(\lambda(N)) = \pi(x_0)$ for some $x_0$).*

*Proof.* For $N$ of the prescribed form, $\lambda(N) = \operatorname{lcm}[2P_1, 2Q_1] = 2P_1Q_1$, and $\lambda(\lambda(N)) = \operatorname{lcm}[2P_2, 2Q_2] = 2P_2Q_2$. It is easy to see that $\lambda(\lambda(N)/2) = \lambda(\lambda(N))$, so $\operatorname{ord}_{\lambda(N)/2}(2) | \lambda(\lambda(N))$. Therefore, $\operatorname{ord}_{\lambda(N)/2}(2) | 2P_2Q_2$.

Assume to the contrary that $\operatorname{ord}_{\lambda(N)/2} 2 \neq 2P_2Q_2$. Then either $\operatorname{ord}_{\lambda(N)/2}(2) = P_2Q_2$ or $\operatorname{ord}_{\lambda(N)/2}(2) | 2P_2$ or $\operatorname{ord}_{\lambda(N)/2}(2) | 2Q_2$. Without loss of generality, we may assume that $\operatorname{ord}_{\lambda(N)/2} 2 | 2P_2$ or $\operatorname{ord}_{\lambda(N)/2}(2) = P_2Q_2$.

[5] Roughly three fourths of all special numbers of the prescribed form satisfy this additional condition (that 2 is a quadratic residue with respect to at most one of $P_1$ and $Q_1$). The condition is needed: for example, the special number in prescribed form, $N = 719 * 47$, fails this condition (for this $N$, $\operatorname{ord}_{\lambda(N)/2}(2) = \lambda(\lambda(N))/2$).

*Case* 1. $\text{ord}_{\lambda(N)/2}\, 2 | 2P_2$.

Then $2^{2P_2} \equiv 1 \bmod (\lambda(N)/2) \equiv 1 \bmod P_1Q_1$.

Therefore $2^{2P_2} \equiv 1 \bmod Q_1$.

But $2^{2Q_2} \equiv 1 \bmod Q_1$ since $Q_1 = 2Q_2 + 1$, by Fermat's Little Theorem.

Therefore $2^{\gcd(2P_2, 2Q_2)} \equiv 1 \bmod Q_1$.

Therefore $2^2 \equiv 1 \bmod Q_1$. This is a contradiction (since $Q_2 \geqq 3$ and therefore $Q_1 \geqq 7$).

*Case* 2. $\text{ord}_{\lambda(N)/2}(2) = P_2Q_2$. Then $2^{P_2Q_2} \equiv 1 \bmod P_1Q_1$, which implies $2^{P_2Q_2} \equiv 1 \bmod Q_1$, which implies $2^{Q_2} \not\equiv -1 \bmod Q_1$ since $P_2$ is odd. Therefore, $2^{(Q_1-1)/2} \not\equiv -1 \bmod Q_1$. Therefore, 2 is a quadratic residue with respect to $Q_1$. Similarly, 2 is a quadratic residue with respect to $P_1$. Contradiction. QED

*Open question.* Let $\pi_b(x_0)$ be the period of the sequence $b_0 b_1 \cdots$ produced by the $x^2 \bmod N$ generator with input $(N, x_0)$. Then $\pi_b(x_0) | \pi(x_0)$. What is the exact relation between $\pi_b(x_0)$ and $\pi(x_0)$? Are they generally equal?

**9. Algorithms for determining length of period and random accessing.** The following two theorems provide algorithms for determining

(1) $\pi(x_0)$, the period of the $x^2 \bmod N$ sequence that begins with $x_0$, and

(2) the $i$th element $x_i$.

These will be useful in the cryptographic applications.

THEOREM 9. *There exists an efficient algorithm A which, when given any N of the prescribed form,*[6] $\lambda(N)$, $\lambda(\lambda(N))$ *AND the factorization of* $\lambda(\lambda(N))$, *efficiently determines the period* $\pi(x_0)$ *of any quadratic residue* $x_0$ *in* $Z_N^*$, *i.e.,* $A[N, \lambda(N), \lambda(\lambda(N)),$ *factorization of* $\lambda(\lambda(N)), x_0] = \pi(x_0)$.

*Proof.* Let $\pi = \pi(x_0)$.

Recall that

(1) $x_i = (x_0)^{2^i} \bmod N$ and $x_\pi = (x_0)^{2^\pi} \bmod N = x_0$.

(2) $\pi | \lambda(\lambda(N))$ (by Theorem 6).

Therefore, $(x_0)^{2^{\lambda(\lambda(N))}} \bmod N = x_0$.

It follows that $(x_0)^{2^{\lambda(\lambda(N)) \bmod \lambda(N)}} \bmod N = x_0$ (by Carmichael's extension of Euler's theorem, $\alpha^{\lambda(N)} \equiv 1 \bmod N$ if $\gcd(\alpha, N) = 1$; therefore $x_0^{\lambda(N)} \equiv 1 \bmod N$; therefore $x_0^{2^{\lambda(\lambda(N))} + k\lambda(N)} \bmod N = x_0$).

Therefore, from knowledge of $\lambda(N)$, $\lambda(\lambda(N))$, and the factorization of $\lambda(\lambda(N))$, one can efficiently determine $\pi$: look for the largest $d | \lambda(\lambda(N))$ such that $(x_0)^{2^{\lambda(\lambda(N))/d \bmod \lambda(N)}} \bmod N = x_0$. Then $\pi = \lambda(\lambda(N))/d$. QED

THEOREM 10a. *There exists an efficient deterministic algorithm A such that given* $N, \lambda(N)$, *any quadratic residue* $x_0$ *in* $Z_N^*$, *and any positive integer i, A efficiently computes* $x_i$, *i.e.,*

$$A[N, \lambda(N), x_0, i] = x_i.$$

*Proof.* $x_i = x_0^{2^i \bmod \lambda(N)} \bmod N$.

The number of steps to compute $x_i$ in this fashion, given $N$, $\lambda(N)$, $x_0$ and $i$, is $O(|N|^{2+\varepsilon})$ using fast multiplication. QED

THEOREM 10b. *There exists an efficient deterministic algorithm A such that given any N of the prescribed form,* $\lambda(N)$, *any quadratic residue* $x_0$ *in* $Z_N^*$, *and any positive integer i, A efficiently computes* $x_{-i}$ (*note the negative subscript*), *i.e.,* $A[N, \lambda(N), x_0, i] = x_{-i}$.

*Proof.* [Miller] has shown how to efficiently factor $N = P \cdot Q$ given $\lambda(N)$. The proof of Theorem 3 shows that $\sqrt{x_0} \bmod P = \pm x_0^{(P+1)/4}$. Exactly one of these two

[6] $N = P * Q$, where $P$, $Q$ are primes congruent to 3 mod 4.

roots is a quadratic residue since $-1$ is *not* a quadratic residue mod $P$ for $P \equiv 3 \bmod 4$. Therefore, $x_{-1} \bmod P = x_0^{P+1/4}$ or $-x_0^{P+1/4}$. Similarly, $x_{-2} \bmod P = \pm x_0^{(P+1/4)^2}$ (since $(-1)^{P+1/4} = \pm 1$). Continuing, $x_{-i} \bmod P = \pm x_0^{(P+1/4)^i} \bmod P = \pm x_0^{(P+1/4)^i \bmod (P-1)}$ mod $P$, which can be computed efficiently. From $x_{-i} \bmod P$ and $x_{-i} \bmod Q$, The Chinese Remainder Theorem enables one to efficiently compute $x_{-i}$. QED

Conversely, the following theorem asserts that an algorithm that knows the period, $\pi$, and for any $i$ can obtain the $i$th element $x_i$ of the sequence $x_0, x_1, \cdots$ obtained by squaring mod $N$ can factor $N$.

THEOREM 11. *Let $O$ denote an oracle such that $O(N, x_0, i) = \langle \pi, x_i \rangle$, where $\pi = \pi(x_0)$. There is an efficient probabilistic algorithm $A^O$ such that $A^O(N) = P$ or $Q$, for $N = P * Q$.*

*Proof.* The algorithm $A^O$ works as follows:

Search at random for $y \in Z_N^*$ such that $(y/N) = -1$ (half the elements of $Z_N^*$ have Jacobi symbol $-1$ with respect to $N$). Set $x_0 = y^2 \bmod N$. Ask the Oracle for $\pi$, then for $x_{\pi-1}$ (recall: $x_\pi = x_0$). Then $y^2 = (x_{\pi-1})^2 = x_\pi = x_0 \bmod N$. But $y \neq \pm x_{\pi-1}$ since $(y/N) = -1$ and $(\pm x_{\pi-1}/N) = +1$. Therefore, $\gcd(y + x_{\pi-1}, N) = P$ or $Q$ (by elementary number theory). QED

*Open question.* Can an algorithm use an oracle that outputs just $x_i$—namely, $O(N, x_0, i) = x_i$—to factor $N$?

*Open question.* Can an algorithm use an oracle that outputs just $\pi$—namely, $O(N, x_0) = \pi$—to factor $N$?

**10. Applications.** (1.1) The $1/P$ generator (base 2) is useful for constructing (generalized) de Bruijn sequences. These have applications, for example, in the design of radar for environments with extreme background noise [Golomb]. We believe there may be additional interesting applications making use of properties identified in this paper, particularly the property that from $2|P|+1$ but *not* $|P|-1$ quotient digits, one can infer the sequence backwards and forwards. For example, one could split a key, $P$, between two parties—by giving $|P|$ successive quotient digits to each so that together they have $2|P|$ successive digits. Neither party alone would have the slightest information *which* prime, $P$, was key, but cooperatively they could determine $P$ efficiently.

(1.2) Maximum-length shift-register sequences (which are closely related to the $1/P$ *generator*) are used for encryption of messages [Golomb]. We view the inference procedure given here as yet another step toward breaking such crypto-systems.

(2.1) The $x^2 \bmod N$ sequence can be used for *public-key cryptography*: Alice can enable Bob to send messages to her (over public channels) that only she can read. Alice constructs and publicizes a number $N_A$, her *public key*, with the prescribed properties: $N_A = P_A * Q_A$ where $P_A$ and $Q_A$ are distinct equal length randomly chosen primes both congruent to 3 mod 4. She keeps private the primes $P_A$ and $Q_A$, her *private key.*

Bob encrypts: Suppose Bob wants to send a $k$-bit message $\vec{m} = (m_1, \cdots, m_k)$, where $k = \text{poly}(|N_A|)$, to Alice. Using Alice's public key, Bob constructs a *one-time pad*: he selects an integer $x_0$ from $Z_{N_A}^*$ at random, squares it mod $N_A$ to get a quadratic residue $x_1$, and uses the $x^2 \bmod N$-generator with input $(N_A, x_1)$ to generate the one-time pad $\vec{b} = (b_1, \cdots, b_k)$. Bob then sends BOTH the encrypted message, $\vec{m} \oplus \vec{b} = (m_1 \oplus b_1, \cdots, m_k \oplus b_k)$, AND $x_{k+1}$ to Alice over public channels, where $\oplus$ is the exclusive-or.

Alice decrypts: From her knowledge of $P_A$ and $Q_A$, her private key, Alice has enough information to efficiently compute $x_k, x_{k-1}, \cdots, x_1$ from $x_{k+1}$ by backwards

jump (Theorem 3). From that, she reconstructs the one-time pad $\vec{b}$ and, by $\oplus$-oring $\vec{b}$ with the encrypted message, decrypts the message, $\vec{m}$.

Anyone who can reconstruct (i.e., guess with some advantage) even one bit of $\vec{m}$ from knowledge of $k$ and $x_{k+1}$ can thereby obtain (guess with some advantage) a bit of the one-time pad $\vec{b}$. This is impossible (by the quadratic residuacity assumption and the following theorem) if $\vec{m}$ is a randomly selected message.

THEOREM 12 (stronger version of claim 3). *Let* poly *be a polynomial. Let* $0 < \delta < 1$. *Let t be a positive integer. Then for all but a* $\delta$*-fraction of numbers N of the prescribed type*[8], *the factors of N are necessary—assuming they are necessary for deciding quadratic residuacity of x in* $Z_N^*(+1)$*—to have even an* $\varepsilon$*-advantage,*[7] $\varepsilon = 1/|N|^t$, *in guessing in* poly*-time any pair* $(l, b_l)$ *(i.e., any bit* $b_l$ *and its location l in the sequence* $b_1, \cdots, b_k$*),* $1 \leqq l \leqq k = \text{poly}(|N|)$ *given N and* $x_{k+1}$, *where* $b_l = \text{parity}(x_l)$.

*Proof.* Assume to the contrary that the probabilistic poly-time procedure **P** has an $\varepsilon$-advantage in guessing a pair. This **P** can be used to obtain a probabilistic poly-time procedure that has an $\varepsilon/\text{poly}(|N|)$-advantage in deciding quadratic residuosity of a *randomly-chosen* $x \in Z_N^*(+1)$: Select $l$, $1 \leqq l \leqq \text{poly}(|N|)$, *at random* with the uniform probability distribution, set $x_{l+1} = x^2 \bmod N$, and generate $x_{k+1}$. Compute $\mathbf{P}[N, x_{k+1}]$. The chances are $1/\text{poly}(|N|)$ that $\mathbf{P}[N, x_{k+1}] = (l, b)$ for the above-chosen $l$ and *some* $b$. If so, then guess that $x$ is a quadratic residue if and only if parity $(x) = b$. If not, toss a fair coin to decide quadratic residuacity of $x$. The advantage (in guessing quadratic residuacity of $x$) will be $\varepsilon/\text{poly}(|N|)$. QED

(2.2) Having constructed a number $N_A = P_A \cdot Q_A$ with the prescribed properties, Alice can compute $\lambda(N)$ and use it, by Theorem 10a, to quickly compute $x_i = x_0^{2^i \bmod \lambda(N)} \bmod N$ (for any $x_0 \in QR_N$). This means she can use word $i$ as address to retrieve word $x_i$ or bit $b_i$ efficiently—as if the $x^2 \bmod N$ generator were a random access memory that is storing a pseudo-random sequence. [Brassard] has suggested applications, e.g., to the construction of unforgeable subway tokens, where this jumping ahead is desirable.

(2.3) Cryptographically secure pseudo-random sequence generators (such as the $x^2 \bmod N$ generator) may be viewed as *amplifiers* of randomness (short random strings are amplified to make long pseudo-random strings).

(2.4) One often uses pseudo-random sequences (rather than random sequences) because they are reproducible [von Neumann]. For the pseudo-random sequences produced by the $x^2 \bmod N$ generator, one has only to store a short seed in order to reproduce a long sequence; one does not have to store the entire random sequence.

**11. Brief history relevant to this paper.** W. Diffie and M. Hellman [Diffie–Hellman] first introduce the notion of a trapdoor function and public-key cryptography.

R. Rivest, A. Shamir and L. Adleman [Rivest–Shamir–Adleman] propose the first concrete example (and implementation to public-key cryptography) of a trapdoor function relying on (but not yet proved equivalent to) a number theoretic conjecture (which they propose) that factoring is hard. The RSA trapdoor function is $x^s \bmod N$ (where $N = P \cdot Q$, $P$, $Q$ are distinct odd primes and $\gcd(s, \varphi(N)) = 1$). Later [Shamir–Rivest–Adleman] utilize a private-key commutative function in their solution to the problem of mental poker posed by R. Floyd.

---

[7] DEFINITION. A probabilistic poly-time procedure $\mathbf{P}[N, x_{k+1}]$ has an $\varepsilon$*-advantage for N in guessing a pair* $(l, b_l)$, $1 \leqq l \leqq k = \text{poly}(|N|)$ (given arbitrary $x_{k+1}$ selected uniformly from $QR_N$) if and only if

$$\left( \frac{\sum_{x_{k+1} \in QR_N} \text{Prob}\left(V_{l=1}^{k} \{\mathbf{P}[N, x_{k+1}] = (l, \text{parity}(x_l))\}\right)}{(\varphi(N))/4} \right) \geqq (1/2) + \varepsilon.$$

M. O. Rabin [Rabin '79] introduced for his signature scheme the many-one trapdoor $x^2 \bmod N$ (where $N = P \cdot Q$ for distinct odd primes $P$, $Q$), which he proves is as hard to invert as factoring.

M. Blum [Blum] for his coin-flipping algorithm first chose $P \equiv Q \equiv 3 \bmod 4$ to construct a trapdoor (the 3 mod 4 trapdoor) $x^2 \bmod N$ (as hard to invert as factoring) which is $1-1$ on the quadratic residues mod $N$.

S. Goldwasser and S. Micali [Goldwasser–Micali] use these properties (of the $x^2 \bmod N$ trapdoor and the 3 mod 4 trapdoor) and the quadratic residuacity assumption which they first propose, to construct a protocol for mental poker and an encryption scheme that hides partial information. This directly addresses the problem pointed out by R. Lipton [Lipton] that partial information can be preserved and transmitted by trapdoor functions (e.g. the set of quadratic residues is invariant under the RSA function) giving rise to an advantage, and enabling trapdoors to be inverted on certain message spaces.

A. Shamir [Shamir] proposed the first example (based on RSA) of a cryptographically strong (i.e. polynomial-time unpredictable) pseudo-random sequence generator.

M. Blum and S. Micali [Blum–Micali] present general conditions on predicates that will ensure a cryptographically strong generator. Using these conditions and the Discrete Logarithm Conjecture they construct cryptographically strong sequences of pseudo-random bits.

A. Yao [Yao], in his foundational paper on complexity based information theory, constructs a "perfect" pseudo-random sequence generator on the very general assumption that there exists a so-called "stable" one-way function.

Our $x^2 \bmod N$ generator is based directly on a 3 mod 4 trapdoor and the *QRA*. We believe that the 3 mod 4 scenario, because of its nice mathematical properties (e.g. Lemma 1) will continue to lead to fruitful applications. We also believe that an in-depth analysis of sequences produced by unpredictable pseudo-random sequence generators, as begun in this paper, will provide useful information concerning the nature of these generators, and lead to insights about the number theoretic assumptions that have been made.

**Acknowledgments.** We thank Silvio Micali for pointing us to the literature on de Bruijn sequences, and for his numerous helpful and encouraging suggestions. We are grateful to a number of people for valuable discussions on this work, including G. Brassard, S. Even, A. Lempel, L Levin, J. Plumstead, M. O. Rabin, D. Rich, S. Smale, R. Solovay and A. Yao. Umesh Vazirani provided a necessary ingredient for our proof of Theorem 5 [Yao]; Rene Peralta did the same for Theorem 10b.

*Note added in proof.* The assertion "modulo the *QRA*" and/or its equivalent in Theorems 4, 5 and 12 can be replaced by "modulo the assumption that factoring is hard". This is a consequence of the main theorem in W. ALEXI, B. CHOR, O. GOLDREICH AND C. P. SCHNORR, *RSA/Rabin bits are* $(\frac{1}{2})+(1/\text{poly}(\log N))$ *secure*, IEEE 25th Symposium on Foundations of Computer Science, 1984, pp. 449–457.

## REFERENCES

[1] L. ADLEMAN, *On distinguishing prime numbers from composite numbers*, Proc. 21st IEEE Symposium on Foundations of Computer Science, 1980, pp. 387–408.

[2] E. BACH, *How to generate random integers with known factorization*, submitted for publication.

[3] P. BILLINGSLEY, *Ergodic Theory and Information*, John Wiley, New York, 1965.

[4] M. BLUM, *Coin flipping by telephone*, in Proc. IEEE Spring COMPCON, 1982, pp. 133–137.

[5] M. BLUM AND S. MICALI, *How to generate cryptographically strong sequences of pseudo random bits*, IEEE 23rd Symposium on the Foundations of Computer Science (1982), pp. 112–117.

[6] G. BRASSARD, *On computationally secure authentication tags requiring short secret shared keys*, in Advances in Cryptology, Proc. of Crypto 82, ed. D. Chaum, R. L. Rivest and A. T. Sherman, Plenum Press, New York, 1983, pp. 79–86.
[7] L. DICKSON, *History of the Theory of Numbers*, Chelsea Pub. Co., 1919 (republished 1971).
[8] W. DIFFIE AND M. HELLMAN, *New directions in cryptography*, IEEE Trans. Inform. Theory, IT-22 (Nov. 1976), pp. 644–654.
[9] S. EVEN, *Graph Algorithms*, Computer Science Press, Potomac, MD, 1979.
[10] C. G. GAUSS, *Disquisitiones Arithmeticae*, 1801; reprinted in English transl. by Yale Univ. Press, New Haven, CT, 1966.
[11] S. GOLDWASSER AND S. MICALI, *Probabilistic encryption and how to play mental poker keeping secret all partial information*, 14th STOC, 1982, pp. 365–377.
[12] S. GOLOMB, *Shift Register Sequences*, Aegean Park Press, 1982.
[13] J. HOPCROFT AND J. ULLMAN, *Formal Languages and Their Relation to Automata*, Addison-Wesley, Reading, MA, 1969.
[14] M. KAC, *What is randomness?*, American Scientist, 71 (August 1983), pp. 405–406.
[15] D. KNUTH, *The Art of Computer Programming: Vol. 2, Seminumerical Algorithms*, Addison-Wesley, Reading, MA, 1981.
[16] W. LEVEQUE, *Fundamentals of Number Theory*, Addison-Wesley, Reading, MA, 1977.
[17] R. LIPTON, *How to cheat at mental poker*, Univ. California, Berkeley, preliminary report, August 1979.
[18] G. MILLER, *Riemann's hypothesis and tests for primality*, Ph.D. thesis, Univ. California, Berkeley, 1975.
[19] J. PLUMSTEAD, *Inferring a sequence generated by a linear congruence*, IEEE 23rd Symposium on Foundations of Computer Science, 1982, pp. 153–159.
[20] S. POHLIG AND M. HELLMAN, *An improved algorithm for computing logarithms over GF(p) and its cryptographic significance*, IEEE Trans. Inform. Theory, IT-24 (1978), pp. 106–110.
[21] M. O. RABIN, *Digital signatures and public-key functions as intractable as factorization*, MIT/LCS/TR-212 Tech. memo, Massachusetts Institute of Technology, 1979.
[22] ———, *Probabilistic algorithm for testing primality*, J. Number Theory, 12 (1980), pp. 128–138.
[23] R. RIVEST, A. SHAMIR AND L. ADLEMAN, *A method for obtaining digital signatures and public key cryptosystems*, Comm. ACM, 21 (1978), pp. 120–126.
[24] A. SHAMIR, R. RIVEST AND L. ADLEMAN, *Mental poker*, in The Mathematical Gardner, D. Klarner, ed., Wadsworth, New York, 1981, pp. 37–43.
[25] A. SHAMIR, *On the generation of cryptographically strong pseudo-random sequences*, ICALP, 1981.
[26] D. SHANKS, *Solved and Unsolved Problems in Number Theory*, Chelsea, New York, 1976.
[27] R. SOLOVAY AND V. STRASSEN, *A fast Monte-Carlo test for primality*, this Journal, 6 (1977), pp. 84–85.
[28] J. VON NEUMANN, *Various techniques used in connection with random digits*, Collected Works, vol. 5, Macmillan, New York, 1963, pp. 768–770.
[29] A. YAO, *Theory and applications of trapdoor functions*, IEEE 23rd Symposium on Foundations of Computer Science, 1982, pp. 80–91.