COMPARISON OF TWO PSEUDO-RANDOM NUMBER GENERATORS[1]

Lenore Blum[2]

Dept. of Mathematics
and Computer Science
Mills College,
Oakland, Ca., and
Dept. of Mathematics
U.C. Berkeley

Manuel Blum[3]

Dept. of Electrical
Engineering and
Computer Sciences
University of California
Berkeley, CA

Michael Shub[4]

Dept. of Mathematics
Queens College
Flushing, NY
and Graduate Center
of CUNY, NY

1. Introduction

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 only require of such generators 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 random sequences is their unpredictability. Pseudo-random sequences should also be unpredictable to computers with feasible resources. To begin to capture this notion, we require the following definition:

DEFINITION: A pseudo-random sequence generator is ***polynomial-time unpredictable*** (unpredictable to the right, unpredictable to the left) or ***cryptographically secure*** [Shamir, Blum-Micali, Yao] if and only if the sequences it generates are not predictable (to the right, to the left) in polynomial time: i.e., given a piece of

[1] The final version of this paper contains the proofs of all theorems discussed here. It will appear in the SIAM Journal of Computing.

[2] This work was supported in part by the Letts-Villard Chair, Mills College.

[3] This work was supported in part by NSF grant MCS 82-04506.

[4] This work was supported in part by NSF grant MCS 82-01267.

sequence that has been produced by such a generator, but with any element (the rightmost element, the leftmost element) deleted from that piece, one can roughly do no better in guessing in polynomial time what the missing element is than by flipping a fair coin.

This notion of unpredictable is the correct one [Yao]: sequences produced by generators possessing any one of the unpredictability properties pass all polynomial time statistical tests. That is to say, these sequences cannot be distinguished by polynomial time statistical tests (with more than a negligible advantage) from sequences produced by successive flips of a fair coin.

2. 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 *mod* 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 residuosity problem, respectively). **But only the second is cryptographically secure.**

More specifically,

THEOREM 2 - Problem 4, section 6, ($1/P$): Any sequence produced by the $1/P$ generator is completely inferable; that is, given a small piece of the sequence, one can quickly infer the "seed" and efficiently extend the given piece of sequence backwards and forwards.

On the other hand,

THEOREM 4, section 7, (x^2 *mod* N): The x^2 *mod* N generator is polynomial-time unpredictable. The sequences it generates are *provably secure:* they pass all polynomial-time statistical tests.

The $1/P$ generator has been well studied [Dickson]; its distribution properties relate to shift register sequences [Golomb]. Our observations concerning its strong inference properties, we believe, are new and surprising. The x^2 *mod* N generator is new, a simplification of a generator proposed by A. Yao. Its strong security properties derive from complexity based number theoretic assumptions and arguments [Blum, Blum&Micali, 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 *mod* N generator has applications to public-key cryptography and to efficient computation, i.e., to converting probabilistic polynomial-time algorithms to deterministic almost-polynomial-time algorithms (see section 10, Applications).

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

3. NOTATION

Throughout this paper, $x \; mod \; N$ denotes the least non-negative integer remainder upon dividing x by N (rather than denoting the residue class mod N).

We use "$x^2 \; mod \; N$ generator" to denote a pseudo-random sequence generator, whereas "$x^2 \; mod \; N$" denotes the remainder upon dividing x^2 by N. A similar distinction is made between "$1/P$ generator" and the string of bits "$1/P$".

We let $|N|_b$ denote the length of N when N is expanded base b, and simply write $|N|$ when the base is clear. Note that $|N|_b = \lfloor 1+\log_b N \rfloor$.

4. THE 1/P GENERATOR

DEFINITION (the $1/P$ generator): Let P be an odd prime. Let b be a generator (i.e., primitive root) of the multiplicative group $Z_P^* = \{1, 2, \ldots, P-1\}$ of integers mod P.[5] The pseudo-random sequence generated by the $1/P$ ***generator*** with input (b, P) is the sequence of b-ary quotient digits that immediately follows the decimal point when $1/P$ is expanded to base b. We denote it by $q_1 q_2 \ldots$. More generally, let r_0 be any integer in the range $0 < r_0 < P$. The pseudo-random sequence generated by the $1/P$ generator with input (b, P, r_0) is the sequence of digits obtained upon dividing r_0 by P. The expansion of $1/P$ (and more generally, r_0/P) is periodic with period P-1: $1/P = q_1 q_2 \cdots q_{P-1} q_1 \cdots$

EXAMPLE: Let $b = 10$ and $P = 7$. This b is a primitive root mod P. The pseudo-random sequence generated by the $1/P$ generator with input (10, 7) is 142857142... since 1/7 = .142857142....

5. THE $x^2 \; mod \; N$ GENERATOR

DEFINITION (the $x^2 \; mod \; N$ generator): Let $N = P*Q$ be a product of two distinct primes both congruent to 3 mod 4. Let x_0 be any quadratic residue in $Z_N^* = \{\text{integers } x \mid 0 < x < N \text{ and } \gcd(x,N) = 1\}$. Here, N is called the ***parameter*** and x_0 the ***seed***. The pseudo-random sequence generated by the ***x^2 mod N generator*** with input (N, x_0) is the sequence of bits $b_0 b_1 \cdots$ obtained by setting $x_{i+1} = x_i^2 \; mod \; N$ and extracting the bit $b_i = parity(x_i)$. This sequence is periodic with period that is usually equal to $\lambda(\lambda(N))$ (see section 8 for the definition of λ and clarification of "usually"). We also note that the equality $x_i = x_0^{2^i} \; mod \; N = x_0^{2^i \; mod \; \lambda(N)} \; mod \; N$ enables us to efficiently compute the ith sequence element, given x_0, N and $\lambda(N)$.

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

6. THE ASSUMPTIONS

Our main results about cryptographic security follow from assumptions concerning the intractability of certain number-theoretic problems by proba-

[5]E. Artin's conjecture states that every integer b which is not -1 or a square is a primitive root for .3739... of all n-bit primes, asymptotically as the length n of the primes goes to infinity -- see [Shanks, p.81].

bilistic polynomial-time procedures. These results can be viewed as assertions concerning the Turing machine complexity (equivalence) of certain hard problems. 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 generator for Z_P^*. The function $f_{b,P} : Z_P^* \dashrightarrow Z_P^*$ defined by $f_{b,P}(x) = b^x \bmod P$ is a permutation of Z_P^* that is computable in $O(|P|^3)$-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 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 there is a fixed fraction of time that the discrete logarithm problem cannot be solved efficiently.) Let $\mathbf{P}[b, P, y]$ be a (probabilistic) procedure for solving the discrete logarithm problem. Let $0 < \varepsilon < 1$ be a fixed constant. Let *poly* be a fixed polynomial. Then for all sufficiently large n, for all but ε-fraction of n-bit primes P, for all generators b of Z_P^*, and for at least ε-fraction of numbers $y \in Z_P^*$, $\mathbf{P}[b, P, y]$ takes more than $poly(n)$ (expected) time to output x (the particular x such that $b^x \bmod P = y$).

(2) THE QUADRATIC RESIDUOSITY PROBLEM [Gauss]: Let N be a product of two distinct odd primes. Let Z_N^* be the group of integers x, $1 \le x \le N$ and $gcd(x, N) = 1\}$, under multiplication mod N. 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 residuosity problem*** with parameters N and x consists in deciding, for x in $Z_N^*(+1)$, whether or not x is a quadratic residue. A probabilistic procedure $\mathbf{P}[N, x]$ solves the quadratic residuosity problem for a number N, N = a product of two distinct odd primes, and for $x \in Z_N^*(+1)$ if and only if it correctly decides whether or not x is a quadratic residue mod N (i.e., $\mathbf{P}[N, x] = 1$ if and only if x is a quadratic residue mod N).

THE QUADRATIC RESIDUOSITY ASSUMPTION: (This asserts that there is a fraction of time that the quadratic residuosity problem cannot be solved efficiently.) Let $poly(\ )$ be a polynomial. Let t be a positive integer. Let $\mathbf{P}[N, x]$ be any (probabilistic) *poly*-time procedure which, on inputs N, x, each of length n, outputs 0 or 1. Then for n sufficiently large and for all but $1/n^t$ fraction of numbers N of length n, N a product of two distinct odd primes, the probability that $\mathbf{P}$ decides correctly whether x is a quadratic residue mod N -- for N fixed and x selected uniformly from among all elements of $Z_N^*(+1)$ -- is less than $1 - 1/n^t$.

7. THE 1/P GENERATOR IS INFERABLE

Let P and b be relatively prime positive integers and r_0 an integer in the range $0 < r_0 < P$. Denote the expansion of r_0/P to base b by

$$r_0/P = .q_1q_2q_3 \cdots \tag{1}$$

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

$m \geq 0$,

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

where

$$0 < r_m = b^m r_0 \; mod \; P < P \qquad (3)$$

and

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

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: $|r_m| \leq |P|$. Recall from the definition of the $1/P$ generator in section 3 that, for P prime and b a primitive root mod P, eq. 1 defines the pseudo-random sequence generated by the $1/P$ generator with input (b, P, r_0).

There are several reasons one might consider the $1/P$ generator a good pseudo-random sequence generator: the sequences produced have long periods and nice distribution properties (Theorem 1 below). 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 $r_m = b^m \; mod \; P$, so m is the discrete logarithm of $r \; mod \; P$. It follows (Theorem 2, problem 1) that, given a string of quotient digits $q_{m+1} q_{m+2} \cdots q_{m+k}$ $(k \leq poly(|P|))$, it is hard in general to find its location in the sequence. Thus, these strings appear to be "well mixed."

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 sequence, 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 extracted from $1/P$, 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 $\geq 1 - 1/2^{n-1}$) been generated using the $1/P$ generator.

To lead up to Theorem 1, we consider the following type 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 $q_1 q_2 \cdots$ of b-ary digits (i.e., $0 \leq q_i < b$ 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 $b \in \{1,2,\ldots,P-1\}$ be a primitive root $mod \; P$. Let $r_0 \in \{1,2,\ldots,P-1\}$. Then the pseudo-random sequence generated by the $1/P$ generator with input (b, P, r_0) 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 $1/P$) is periodic with period P-1, the remainders in any period are distinct, and $\{r_m \mid 1 \le m \le P-1\} = \{1,2,\ldots,P\text{-}1\}$.

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

$$\{r_m / P \mid 1 \le m \le P-1\} = \{1/P, 2/P, \ldots, (P-1)/P\}. \qquad (5)$$

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 $0 \le m_1 < m_2 < P-1$ 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 $1 \le m \le P-1$ if and only if (by eq. 5) it appears as an initial string in the expansion of k/P for some $1 \le k \le P-1$. 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, $0 \le l < b^s$. Since $1/P < 1/b^{|P|-1}$, there is for each l, at least one k, $1 \le k \le P-1$ 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, $1 \le k \le P-1$ 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 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, $poly(\ )$, and hold it fixed.
INPUT: P, b, remainder r_m, positive integer $k \le poly(|P|)$.
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 $1,2,\dots,b$ (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. $r_m \geq P/b$).
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 $k \leq 2|P|+1)$[6].
OUTPUT: P; r_m (and hence by Problem 1, q_m and q_{m+k+1}).

PROOF:

To solve problem 1: $r_{m+k} = b^k r_m \; mod \; P$ and $r_{m-1} = b^{-1} r_m \; mod \; P$ 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 eq. 6.)

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

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

To solve problem 3: By eq. 6 with $k = 1$, $b \cdot r_m - r_{m+1} = q_{m+1} \cdot P$ where $0 \leq q_{m+1} < b$. 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 $\dfrac{b \cdot r_m - r_{m+1}}{1}$, $\dfrac{b \cdot r_m - r_{m+1}}{2}, \dots, \dfrac{b \cdot r_m - r_{m+1}}{b-1}$. Select any integer P in the sequence such that P is relatively prime to $1,2,\dots,b$. Such an integer P is unique; for suppose to the contrary that P, Q are two such integers relatively prime to each of $1,2,\dots,b$. 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 $1, 2, \dots, b$, so $gcd(Q, i) = 1$, so $Q | P$, so $Q \leq P$, which is a contradiction.

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

[6] Much as one wants, this result cannot be improved to permit $k = 2|P|$. For example, for $b = 10$ and $P = 97$ ($1/97 = .010309\dots$), the four digits 1030 do not yield P.

b and r_0 are relatively prime to P, it follows (from eq. 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 the first k digits of $\frac{A_i}{B_i}$ 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 $\frac{q_{m+1} \cdots q_{m+k}}{b^k} = 1/a_1 + 1/a_2 + 1/a_3 + \dots 1/a_i + \dots$ has convergents $\frac{A_1}{B_1} = \frac{1}{a_1}$, $\frac{A_2}{B_2} = \frac{a_2}{a_1 a_2 + 1}, \dots, \frac{A_i}{B_i} = \frac{a_i A_{i-1} + A_{i-2}}{a_i B_{i-1} + B_{i-2}}, \dots$. 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 ith Fibonacci number), and therefore in polynomial($|P|$) steps. This solves problem 4.
QED

REMARK: The solution to problem 4 can be viewed as a computational version of the following: for positive integers $b > 1$, k and l, $0 \le l < b^k$, there is at most one integer $P > 1$ with $gcd(b, P) = 1$ and $2P^2 \le b^k$, and at most one integer r with $1 \le r \le P-1$ and $gcd(r, P) = 1$, such that $r/P \in [l/b^k, (l+1)/b^k)$.

EXAMPLE: Let $b = 10$ and $P = 503$. Then P is prime and b is a primitive root mod P, so the $1/P$ generator with input (10, 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 ...

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 seqment of length 6 of the 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 433,399/1,000,000 is 433,399/1,000,000 = 1/2+ 1/3+ 1/3+ 1/1+ 1/16+ 1/6+ 1/1+ 1/1+ 1/358+ ..., and its first five convergents are: 1/2 = .5; 3/7 = .48...; 10/23 = .434...; 13/30 = .43333...; 218/503 = .4333999.... At last, the first 6 digits agree with the segment 433399. So we get $P = 503$ and $r_m = 218$ (and so $r_{m-1} = 10^{-1} \cdot r_m \bmod 503 = 151 * 218 \bmod 503 = 223$). In this way, we can extend the given segment, 433399, forwards and backwards.

8. THE x^2 *mod* N GENERATOR IS UNPREDICTABLE

In this section we elaborate on properties of the x^2 *mod* N pseudo-random sequence generator, and prove 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 $Z_N^* = \{$integers $x \mid 0 < x < N$ 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 *mod* N has four distinct square roots, $\pm x$ *mod* N, $\pm y$ *mod* N. If we also assume, as we shall for the rest of this paper, that $P \equiv Q \equiv 3$ *mod* 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 $n \rightarrow \infty$ [LeVeque], so there are plenty such N.)

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

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.

   Conversely, it follows from more general results of [Plumstead], that there is a polynomial *poly* such that from knowledge of n, and any sequence $x_0, \ldots, x_k$, $k = poly(n)$, generated by x_0 and an unknown N of length n, we can infer in $poly(N)$ - time an N (of length n) that produces this sequence.
2. Given N, the factors of N are necessary and sufficient to efficiently generate the x^2 *mod* 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 residuosity of an x in $Z_n^*(+1)$ -- to have even an ε-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).[7]

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 $P \equiv Q \equiv 3$ *mod* 4, then each quadratic residue mod N has exactly one square root which is a quadratic residue.

PROOF:
Whenever N is a product of two distinct odd primes, every quadratic residue

[7] A more formal statement of claim 3 will appear in the final version of this paper.

mod N has four square roots, $\pm x$ and $\pm y$. Since $N \equiv 1$ mod 4, their Jacobi symbols satisfy $(\frac{+x}{N}) = (\frac{-x}{N})$ and $(\frac{+y}{N}) = (\frac{-y}{N})$. Since P $\equiv$ 3 mod 4, $(\frac{+x}{N}) \neq (\frac{+y}{N})$ (this can easily be proved from the fact that $gcd(x+y, N) = P$ and $gcd(x-y, N) = Q$). Thus $(\frac{+x}{N}) = (\frac{-x}{N}) \neq (\frac{+y}{N}) = (\frac{-y}{N})$. 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 $P \equiv 3 \; mod \; 4$, 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 $(\frac{x}{N}) = -1$. Let $x_0 = x^2 \; mod \; N$ and compute x_{-1}. Then efficiently compute $gcd(x+x_{-1}, N) = P$ or Q. We can sharpen this argument to show [Rabin] 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 $(x_{-1})^2 \; mod \; N = x_0$. 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, $f : x \to x^2 \; mod \; N$, 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 $x_P = \sqrt{x_0} \; mod \; P$ such that $(\frac{x_P}{P}) = +1$, where $\sqrt{x_0} \; mod \; P$ denotes an integer in Z_P^* whose square mod P is x_0 (this computation of $x_P = \sqrt{x_0} \; mod \; P$ can be done efficiently by a deterministic polynomial-time algorithm). Compute $x_Q = \sqrt{x_0} \; mod \; Q$ such that $(\frac{x_Q}{Q}) = +1$. Use the Euclidean algorithm to construct integers u, v such that $P \cdot u + Q \cdot v = 1$, and from that obtain the particular number, $x_N = \pm x_P \cdot Q \cdot v \pm x_Q \cdot P \cdot u = \sqrt{x_0} \; mod \; N$, that is a square root of $x_0 \; mod \; N$, 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: Given a polynomial $poly(\;)$ and $0 < \varepsilon \leq 1/2$, a 0-1 valued probabilistic $poly$-time procedure $\mathbf{P}(\;,\;)$ has an ***ε-advantage for N in guessing (determining) parity*** (of x_{-1} given arbitrary x_0 in QR_N) if and only if given x_0 selected uniformly from QR_N, $Prob[\,\mathbf{P}(N, x_0) = Parity(x_{-1})] > 1/2 + \varepsilon$

In a similar fashion, we can define a procedure having an *ε-advantage for N in guessing quadratic residuosity* (of arbitrary $x \in Z_N^*(+1)$) [Goldwasser-Micali]. In this regard, the $1/2 + \varepsilon$ makes sense since exactly half the elements in $Z_N^*(+1)$ are quadratic residues.

LEMMA 2

An ε-advantage for determining parity (of x_{-1} given quadratic residue x_0) can be converted, efficiently and uniformly, to an ε-advantage for determining quadratic residuosity (of x in $Z_N^*(+1)$).[8]

PROOF

Let $x \in Z_N^*(+1)$ be an element whose quadratic residuosity mod N is to be determined. Set $x_0 = x^2 \textit{ mod } N$. Since $P \equiv Q \equiv 3 \textit{ mod } 4$, the square roots of $x^2 \textit{ mod } N$ that are in $Z_N^*(+1)$ are $\pm x$ (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 $\textit{parity}(x_{-1})$. Therefore, x is a quadratic residue mod N if and only if $x = x_{-1}$ if and only if $\textit{parity}(x) = \textit{parity}(x_{-1})$.

QED

LEMMA 3 (Goldwasser and Micali)

An ε-advantage for determining quadratic residuosity can be amplified as much as we like, uniformly and efficiently.[9]

IDEA OF PROOF

Let $x \in Z_N^*(+1)$ be an element whose quadratic residuosity mod N is to be determined. To this end, select r at random with uniform probability from Z_N^*. Compute $x \cdot r^2 \textit{ mod } N$. [Comment: For $x \in QR_N$, $x \cdot r^2 \textit{ mod } N$ is uniformly distributed over QR_N; for $x \notin QR_N$, $x \cdot r^2 \textit{ mod } N$ is uniformly distributed over $Z_N^*(+1) - QR_N$.] Test each of the resulting numbers, $x \cdot r^2 \textit{ mod } N$, for quadratic residuosity. Taking the majority vote amplifies the advantage as much as one likes.

QED

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 $1/n^t$ of prescribed numbers N of length n). Then convert **P** (Lemma 2) to a probabilistic *poly*' procedure **P'** for determining quadratic residuosity that has an amplified advantage (Lemma 3) of $1/2 - 1/n^{t'}$ (for these same integers N). This contradicts the quadratic residuosity assumption.

Leading up to Theorem 4 we make the following

DEFINITION:

1. A ***predictor*** **P** (,) for the $x^2 \textit{ mod } N$ generator is a probabilistic ***poly***-time procedure that on inputs N, $b_1 \cdots b_k$, with $b_i \in \{0,1\}$ and $k \le \textit{poly}(|N|)$, outputs a 0 or 1.
2. **P** has an *ε-advantage* for N in predicting (to the left) sequences produced by the $x^2 \textit{ mod } N$ generator if and only if $\textit{Prob}\ [\ \mathbf{P}(N, b_1 \cdots b_k) = b_0\ |$ seed x_0 is selected uniformly from QR_N, $k \le \textit{poly}(|N|)$, and $b_1 \cdots b_k$ is the

[8] A more formal statement of Lemma 2 will appear in the final version of this paper.

[9] A more formal statement of Lemma 3 will appear in the final version of this paper.

segment of the sequence generated by the x^2 *mod* N generator with input (N, x_0)] $\geq 1/2 + \varepsilon$.

THEOREM 4

The x^2 *mod* N generator is an unpredictable (cryptographically secure) pseudo-random sequence generator. That is to say, for each probabilistic poly-time predictor P, 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 $1/n^t$ prescribed numbers N of length n).

PROOF:

Suppose we have a predictor for the x^2 *mod* N generator with an ε - advantage for N. This can be converted efficiently and uniformly into a procedure with an ε-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 residuosity with an amplified advantage (Lemma 3) to get a contradiction to the Quadratic Residuosity Assumption.

QED

It follows from Yao's theorem that the sequences produced by the x^2 *mod* N generator pass every probabilistic polynomial-time statistical test. Yao's theorem says, in essence, that the unpredictability property is a universal test for randomness. The *idea* of his argument is as follows. Suppose there were a probabilistic poly-time test T that has an advantage in distinguishing between the pseudo-random sequences produced by an unpredictable generator and truly random sequences of bits. Then, given $k \leq poly(n)$, we can find j (in probabilistic $poly(n)$-time) such that T has an advantage in distinguishing between sequences in $A = \{r_1 \cdots r_j b_0 \cdots b_k\}$ and $B = \{r_1 \cdots r_j r_{j+1} b_1 \cdots b_k\}$, where the $b_0 \cdots b_k$ are sequences produced by the generator, and the $r_1 \cdots r_{j+1}$ are sequences of independent random bits.

We can convert T into a predictor for the generator: Given a sequence $b_1 \cdots b_k$ produced by the generator, we pass a $poly(n)$ sample of sequences of the form $r_1 \cdots r_j 0 b_1 \cdots b_k$ (where the $r_1 \cdots r_j$ are random) through test T. If $b_0 = 0$, then T is more likely to say these sequences belong to A, in which case we predict 0 for b_0. If $b_0 \neq 0$, then the initial segments of these sequences are weighted more heavily on the random side, and thus T is more likely to say they belong to B, in which case we predict 1 for b_0. T's advantage in distinguishing between pseudo-random and random sequences is thus converted into an advantage in predicting b_0 correctly.

REMARK: We can construct another unpredictable generator as follows: recall that since $N \equiv 1$ *mod* 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 $x \geq (N-1)/2$, partitions $Z_N^*(+1)$ in half. Thus we get the following

THEOREM: The modified x^2 *mod* N generator, gotten by extracting the location bit at each stage (instead of parity) is cryptographically secure (modulo the Quadratic Residuosity assumption).

CONJECTURE: The modified x^2 *mod* 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?

9. LENGTHS OF PERIODS (OF THE SEQUENCES PRODUCED BY THE x^2 *mod* N GENERATOR)

What exactly is the period of the sequence generated by the $x^2 \bmod N$ generator? The question arises as soon as one starts to construct examples. Let $\pi(x_0)$ be the period of the sequence $x_0, x_1, x_2, \cdots$. Since the $x^2 \bmod N$ generator is an unpredictable pseudo-random sequence generator, it follows that on the average, $\pi(x_0)$ will be long. In this section we investigate the precise lengths of these periods. 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 $P_1, \ldots, P_k$ are distinct odd primes. Carmichael's λ-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 $\lambda(M) = \mathrm{lcm}[\lambda(2^e), (P_1-1)*P_1^{e_1-1}, \ldots, (P_k-1)*P_k^{e_k-1}]$.
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^*.

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

THEOREM 5:
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: To appear in the final version of this paper.

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

THEOREM 6
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 $ord_{\lambda(N)/2}(2) = \lambda(\lambda(N))$. (Note: this equality frequently holds for prescribed N. See below and Theorem 7.)
2. Choose quadratic residue x_0 so that $ord_N(x_0) = \lambda(N)/2$. (Note: one can always choose a quadratic residue x_0 this way. See below.)
Then $\lambda(\lambda(N)) | \pi(x_0)$ (and therefore $\lambda(\lambda(N)) = \pi(x_0)$).

PROOF: To appear in the final version of this paper.

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 $\frac{\lambda(N)}{2}$ *mod* N is $\Omega\left(\frac{N}{(\ln\ln N)^2}\right)$ (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 $ord_N a$ =

$lcm[P-1,Q-1] = \lambda(N)$. 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 $\varphi(\varphi(P))*\varphi(\varphi(Q))$ elements in Z_N^* of order $\lambda(N)$. But $\varphi(x) > \frac{x}{6\ln\ln x}$ for all integers $x > 2$. Hence $\varphi(\varphi(P))*\varphi(\varphi(Q)) = \varphi(P-1)*\varphi(Q-1) \geq \frac{P-1}{6\ln\ln(P-1)} \cdot \frac{Q-1}{6\ln\ln(Q-1)} \geq \frac{N-P-Q+1}{[6\ln\ln(N-1)]^2} = \Omega\left(\frac{N}{(\ln\ln N)^2}\right)$. The map $x \to x^2 \bmod N$ is 4:1. Therefore, there are at least $\Omega\left(\frac{N}{4(\ln\ln N)^2}\right)$ 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 [cf. 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 $2^n < P < 2^{n+1}$, $2^{n-1} < P < 2^n$, and $2^{n-2} < P < 2^{n-1}$. 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 the Miller-Rabin probabilistic primality test [Miller, Rabin] to select the ones that are special.

THEOREM 7

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.[10] Then $ord_{\lambda(N)/2}(2) = \lambda(\lambda(N))$ (and therefore $\lambda(\lambda(N)) = \pi(x_0)$ for some x_0).

PROOF: To appear in the final version of this paper.

OPEN QUESTION: Let $\pi_b(x_0)$ be the period of the sequence $b_0 b_1 \cdots$ produced by the x^2 *mod* 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?

10. ALGORITHMS FOR DETERMINING LENGTH OF PERIOD AND RANDOM ACCESSING

The following two theorems provide algorithms for determining (1) the period π of x_0 (the x^2 *mod* N sequence that begins with x_0), and

[10] 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, $ord_{\lambda(N)/2}(2) = \lambda(\lambda(N))/2$).

(2) the i^{th} element x_i.
These will be useful in the cryptographic applications.

THEOREM 8
There exists an efficient algorithm A which, when given any N of the prescribed form,[11] $\lambda(N)$, $\lambda(\lambda(N))$ AND the factorization of $\lambda(\lambda(N))$, efficiently determines the period π of any quadratic residue x_0 in Z_N^*, i.e.,
$A[N, \lambda(N), \lambda(\lambda(N))$, factorization of $\lambda(\lambda(N)), x_0] = \pi$, where $\pi = \pi(x_0)$.

PROOF: To appear in the final version of this paper.

THEOREM 9
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: To appear in the final version of this paper.

Conversely, the following theorem asserts that an algorithm that knows the period, π, and the i^{th} element x_i of the sequence $x_0, x_1, \ldots$ obtained by squaring mod N can factor N.

THEOREM 10
Let O denote an oracle such that $O(N, x_0, i) = <\pi, x_i>$, 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: To appear in the final version of this paper.

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 π -- namely, $O(N, x_0) = \pi$ -- to factor N?

11. APPLICATIONS

(1.1) The $1/P$-generator is useful for constructing (generalized) de Bruijn sequences. These have applications, for example, in the design of radar for environments with extreme backround 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.

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

On the other hand, we would be interested to hear of any applications which exploit the property that (1) from a string of quotient digits it is difficult to determine that string's location in the $1/P$-sequence, whereas (2) given a sufficiently long such string, one can nevertheless extend it backwards and forwards.

(2.1) The x^2 *mod* 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 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 an n-bit message $\vec{m} = (m_1, \ldots, m_n)$, where $n = poly(|N_A|)$, 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 *mod* N generator with input (N_A, x_1) to generate the one-time pad $\vec{b} = (b_1, \ldots, b_n)$. Bob then sends BOTH the encrypted message, $\vec{m} \oplus \vec{b} = (m_1 \oplus b_1, \cdots, m_n \oplus b_n)$, AND x_{n+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 $x_n, x_{n-1}, \cdots, x_1$ from x_{n+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 n and x_{n+1} can thereby obtain (guess with some advantage) a bit of the one-time pad $\vec{b}$. This is impossible (by the quadratic residuosity assumption and the following theorem) if $\vec{m}$ is a randomly selected message.

THEOREM (stronger version of claim 3): Suppose N is of the prescribed type. Then the factors of N are necessary -- assuming they are necessary for deciding quadratic residuosity of x in $Z^*_N(+1)$ -- to have even an ε-advantage[12] in guessing in *poly*-time any pair (k, b_k) (i.e., any bit b_k and its location k in the sequence $b_1, \ldots, b_n$), $1 \le k \le n = poly(|N|)$, given N and x_{n+1}.

PROOF: To appear in the final version of this paper.

(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 9, to quickly compute $x_i = x_0^{2^i \bmod \lambda(N)}$ *mod* 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 *mod* 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) As pointed out by Yao, one can convert fast Monte Carlo algorithms into almost-fast deterministic ones by replacing the use of random sequences in

[12] Definition: A *poly*-time procedure P $[N, x_{n+1}]$ has an ***ε-advantage for N in guessing a pair (k, b_k)***, $1 \le k \le n = poly(|N|)$ (given arbitrary x_{n+1} selected uniformly from QR_N) if and only if Prob[P $[N, x_{n+1}] = (k, b_k)$ for some k, $1 \le k \le n$, | x_{n+1} is selected uniformly from QR_N] $\ge 1/2 + \varepsilon$.

such algorithms by sequences produced by a cryptographically secure generator (such as the x^2 *mod* N-generator): If a so-converted Monte Carlo algoirithm were to behave differently (utilizing pseudo-random sequences instead of truly random ones), this algorithm itself would become a test for distinguishing between the two types of sequences (see [Yao] for the many subtle details needed for this argument).

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

(2.5) 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 *mod* 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.

12. ACKNOWLEDGEMENTS

We are grateful to Andy Yao for his ideas and for his encouragement of this work. We thank Silvio Micali for pointing us to the literature on de Bruijn sequences, and for his generally helpful and encouraging suggestions. The broad relevance of the quadratic residuosity assumption to Protocol Design was first pointed out by Shafi Goldwasser and Silvio Micali. We are grateful to a number of people for valuable discussions on this work, including S. Even, A. Lempel, L. Levin, J. Plumstead, and M. O. Rabin.

REFERENCES

[1] L. Adleman, "On Distinguishing Prime Numbers from Composite Numbers," Proc. 21st IEEE Symp. on Found. of Comp. Science (1980), 387-408.

[2] M. Blum, "Coin Flipping by Telephone," in Proc. of IEEE Spring COMPCON (1982), 133-137.

[3] M. Blum and S. Micali, "How to Generate Cryptographically Strong Sequences of Pseudo Random Bits," submitted to FOCS 1982.

[4] G. Brassard, "On computationally Secure Authentication Tags Requiring Short Secret Shared Keys," in Conf. Proc. Crypto 82, 1982.

[5] L. Dickson, "History of the Theory of Numbers," Chelsea Pub. Co., 1919 (republished 1971).

[6] S. Even, "Graph Algorithms," Computer Science Press, 1979.

[7] C. G. Gauss, "Disquisitiones Arithmeticae," 1801; reprinted in English transl. by Yale Univ. Press, 1966.

[8] S. Goldwasser and S. Micali, "Probabilistic Encryption and How to Play Mental Poker Keeping Secret all Partial Information," 14^{th} STOC (1982), 365-377.

[9] S. Golomb, "Shift Register Sequences," Aegean Park Press (1982).

[10] D. Knuth, "The Art of Computer Programming: Seminumerical Algorithms," Vol. 2, Addison-Wesley Pub. Co., 1981.

[11] W. LeVeque, "Fundamentals of Number Theory," Addison-Wesley Pub. Co., 1977.

[12] G. Miller, "Riemann's Hypothesis and Tests for Primality," Ph.D. Thesis, U.C. Berkeley (1975).

[13] J. Plumstead, "Inferring a Sequence Generated by a Linear Congruence," submitted to FOCS 1982.

[14] S. Pohlig and M. Hellman, "An Improved Algorithm for Computing Logarithms over $GF(p)$ and Its Cryptographic Significance," IEEE Trans. on Info. Theory, Vol. It-24, No. 1, (1978), 106-110.

[15] M. O. Rabin, "Probabilistic Algorithm for Tesitng Primality," J. No. Theory, Vol 12 (1980), 128-138.

[16] M. O. Rabin, "Digital Signatures and Public-key Functions as Intractable as Factorization," MIT/LCS/TR-212 Tech. memo, MIT, 1979.

[17] A. Shamir, "On the Generation of Cryptographically Strong Pseudo-Random Sequences," ICALP, 1981.

[18] D. Shanks, "Solved and Unsolved Problems in Number Theory," Chelsea Pub. Co., 1976.

[19] J. von Neumann, "Various Techniques Used in Connection With Random Digits," Collected Works, vol. 5, Macmillan (1963), 768-770.

[20] A. Yao, "Theory and Applications of Trapdoor Functions," submitted to FOCS 1982.