On the Generation of Cryptographically Strong Pseudorandom Sequences

ADI SHAMIR
The Weizmann Institute of Science

This paper shows how to generate from a short random seed a long sequence of pseudorandom numbers which is cryptogrgraphically strong in the sense that knowing some sequence elements cannot possibly help the cryptanalyst to determine other sequence elements. The method is based on the RSA cryptosystem, and it is the first published example of a pseudorandom sequence generator for which such a property has been formally proved.

Categories and Subject Descriptors: E.3 [**Data**]: Data Encryption; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*number theoretic computations*

General Terms: Algorithms, Security

Additional Key Words and Phrases: cryptography, one-time pads, pseudorandom sequences, RSA cryptosystems

1. INTRODUCTION

The simplest and safest cryptosystem is undoubtedly the one-time pad, invented by G.S. Vernam in 1917 (see [1, 2] for more details). Its secret key is a long sequence of randomly chosen bits. A cleartext is encrypted by XORing its bits with an initial segment of the key, and the resultant ciphertext is decrypted by XORing its bits again with the same segment. Each segment is deleted after a single use, so that the key is gradually consumed (see Figure 1). It is easy to show that without knowing the relevant segment of the key, a cryptanalyst cannot determine the cleartext, and thus the system is secure in theory as well as in practice.

The main drawback of one-time pads is the huge key which has to be generated, distributed, and stored by the communicating parties in complete secrecy. In

The research for this paper was partially supported by NSF Grant MCS-8006938.
Author's address: Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel.
This paper was originally submitted for publication in *Communications of the ACM*. The CACM department editor responsible for the paper was Anita K. Jones. The author kindly agreed to publish the paper in this first issue of the *ACM Transactions on Computer Systems.*

cleartexts:	0	1	1	0	0	1	0	0	0	1	1	.	.	.
key:	0	1	0	0	1	0	1	1	0	1	0	.	.	.
ciphertexts:	0	0	1	0	1	1	1	1	0	0	1	.	.	.

Figure 1.

practice, this truly random key is replaced by a pseudorandom running key derived during the encryption/decryption process from an initial seed by a sequence generator such as a shift register with nonlinear feedbacks. The seed (which is a relatively short randomly chosen number describing the initial state of the sequence generator) is the only secret element in this scheme, and it can be used in the encryption of an almost unbounded number of cleartexts.

In order to be cryptographically strong, the pseudorandom running key must be unpredictable. The main problem is to guarantee that even when the cryptanalyst obtains long segments of the running key (by XORing together known cleartext/ciphertext pairs), he should have no knowledge whatsoever about any other segment. Note that the long running key is deterministically generated from the short seed, and thus pure information-theoretic ambiguity arguments become inapplicable once the cryptanalyst obtains enough segments.

The notion of "cryptographic knowledge" of a value is notoriously slippery, and it can be defined in any one of the following forms.

(1) Immediate knowledge—the ability to retrieve the desired value from memory.
(2) Computed knowledge—the ability to compute the desired value within certain time and space complexity bounds.
(3) Partial knowledge—the ability to sharpen the a priori probability distribution of candidate values (e.g., to change a uniform initial distribution into one in which a particular value is the most likely candidate).

The analysis of pseudorandom sequences in this paper is based on definition (2). Consequently, we do not analyze the statistical biases and autocorrelations of our sequences, and we do not consider the possibility of obtaining partial information about some sequence elements (e.g., that their sum is always even). This is admittedly a simplified version of reality, but it is the only one about which we were able to get concrete results. One of the most challenging open problems of cryptography is to develop a unified theory of knowledge that analyzes the information/complexity trade-offs of cryptographic systems—how much information can be gained by investing a given amount of computational resources.

While one can argue heuristically that almost any sequence generated by a complicated multipass randomizing procedure is *likely* to be cryptographically secure under definition (2), the challenge is to generate a sequence which is *provably* secure. At this stage, complexity theory lacks tools for proving the absolute difficulty of computational tasks; thus a more realistic goal is to develop pseudorandom sequence generators which are secure modulo some plausible but unproved assumption (such as $NP \neq P$, the existence of one-way functions, or the difficulty of factoring integers). By clearly identifying the fundamental sources of security and insecurity, such an analysis can put cryptocomplexity on a firmer theoretical basis even when the underlying assumptions are not known to be true.

2. SCHEMES BASED ON ONE-WAY FUNCTIONS

The purpose of this section is to illustrate the trickiness of formal proofs of security by analyzing some simple schemes based on the notion of one-way functions. To simplify the analysis, we axiomatically assume that these functions are permutations on some finite universe U, that they are everywhere easy to compute, and that they are everywhere difficult to invert (more details on this axiomatic approach can be found in [7]).

Given a one-way function f, we can generate a long pseudorandom sequence of elements in U by applying f to some standard sequence of arguments derived from the initial seed S. This sequence can be as simple as

$$S, S + 1, S + 2, \ldots$$

and the cryptanalyst is assumed to know f and the general nature of the sequence, but not S. The values of $f(S + i)$ are considered as indivisible objects rather than bit strings, since we want to avoid problems of partial knowledge about them. Note that unlike the output of shift registers with feedbacks, these sequences do not suffer from error propagation problems, since each element is computed separately from its index and the seed.

The difficulty of extracting S from a *single* value of $f(S + i)$ is guaranteed by the one-way nature of f. However, without further assumption on f, one cannot formally prove that S cannot be extracted from *pairs* of values (such as $f(S), f(S + 1)$). Furthermore, f may be degenerate in the sense that some of its values may be directly computable from other values without computing S first. A simple example which shows that good one-way functions can be misused as sequence generators is supplied by the RSA (Rivest-Shamir-Adelman) encryption function [5]:

$$E_K(M) = M^K (\text{mod } N).$$

This function is believed to be one-way with respect to the key K when the message M and the modulus N are known, but its application to the standard sequence

$$M = 2, 3, 4, 5, 6, \ldots$$

generates the sequence

$$2^K (\text{mod } N), 3^K (\text{mod } N), 4^K (\text{mod } N), \ldots$$

in which the third element is just the square (mod N) of the first element, the fifth element is just the product (mod N) of the first two elements, and so forth. This multiplicative degeneracy makes the sequence insecure even though the secret seed K remains unknown.

A variant of this scheme avoids the problem of multiplicative degeneracy by using the sequence of primes as the standard sequence:

$$M = 2, 3, 5, 7, 11, \ldots.$$

Is the generated sequence secure? We conjecture that it is, but without knowing all the potential degeneracies of the RSA function, we are unable to prove any formal equivalence between the difficulty of computing K from 2^K (mod N) and,

for example, the difficulty of computing $5^K \pmod N$ from $2^K \pmod N$ and $3^K \pmod N$.

Another way (proposed by [4]) in which long sequences may be generated from one-way functions is to iterate their application to the secret seed S. The resultant sequence

$$f(S), f^2(S) = f(f(S)), f^3(S) = f(f(f(S))), \ldots$$

is easy to extend in the forward direction (by applying f), but hard to extend backwards (by applying f^{-1}). If we pick two secret seeds, R and T, generate the two sequences $f^i(R)$ and $f^i(T)$, and XOR pairs of their elements in *opposite directions*

$$f^1(R) \oplus f^n(T), f^2(R) \oplus f^{n-1}(T), \ldots, f^n(R) \oplus f^1(T),$$

we get a sequence which seems to be hard to extend either forwards or backwards. This can be formally proved in the following special case.

LEMMA 1. *If f is a one-way function, then a new element of the sequence cannot be computed from a single known element.*

PROOF. By contradiction. Assume that for some $i \neq j$, $f^i(R) \oplus f^{n-i}(T)$ can be computed from $f^j(R) \oplus f^{n-j}(T)$ for all choices of the unknown seeds R and T. Our goal is to show that given an arbitrary S, $f^{-1}(S)$ can be easily computed, and thus f is not a one-way function.

Without loss of generality, we assume that $i < j$. We pick a random T, and compute $S \oplus f^{n-j}(T)$. Since f is invertible, there is some R (which is hard to compute) such that $S = f^j(R)$. By assumption, from $S \oplus f^{n-j}(T) = f^j(R) \oplus f^{n-j}(T)$, we can compute $f^i(R) \oplus f^{n-i}(T) = f^{i-j}(S) \oplus f^{n-i}(T)$. Knowing T, we can compute $f^{n-i}(T)$ and thus isolate $f^{i-j}(S)$. Since $j - i$ is positive, we can easily apply $fj - i - 1$ times to $f^{i-j}(S)$ to get

$$f^{j-i-1}(f^{i-j}(S)) = f^{-1}(S),$$

and this is the desired result. Q.E.D.

Unfortunately, the XOR operator which scrambles the two sequences together makes it impossible to prove any formal result in more complicated cases. For example, we do not know how to prove that $f^2(R) \oplus f^2(T)$ cannot be computed from $f^1(R) \oplus f^3(T)$ and $f^3(R) \oplus f^1(T)$, if we only assume that f is hard to invert.

In view of these difficulties, it is quite remarkable that for one particular pseudorandom sequence generator based on the RSA function, we can formally prove that no matter how many sequence elements the cryptanalyst gathers, the task of computing one more element remains just as difficult. This scheme is described in the next section.

3. THE PROPOSED SCHEME

The RSA public-key encryption function with modulus N maps the secret cleartext M under the publicly known key K to $M^K \pmod N$. The corresponding decryption function recovers the cleartext by taking the Kth root of the ciphertext (mod N). The cryptographic security of the RSA cryptosystem is thus equivalent

by definition to the difficulty of taking roots mod N. When N is a large composite number with unknown factorization, this root problem is believed to be very difficult, but when the factorization of N (or Euler's totient function $\varphi(N)$) is known and K is relatively prime to $\varphi(N)$, there is a fast algorithm for solving it.

Each pseudorandom sequence generator consists of a modulus N and some standard, easy-to-generate sequence of keys $K_1, K_2, \ldots$, such that $\varphi(N)$ and all the K_i's are pairwise relatively prime. As far as we know, the difficulty of the root problem is determined by the choice of N but not by the choice of the K_i's, and thus almost any segment of odd primes (e.g., 3, 5, 7, 11, ...) can be used as the standard sequence.

In order actually to generate a pseudorandom sequence of values $R_1, R_2, \ldots$, the two parties choose a random seed S and use their knowledge of $\varphi(N)$ to compute the sequence of roots:

$$R_1 = S^{1/K_1} (\text{mod } N), R_2 = S^{1/K_2} (\text{mod } N), \ldots .$$

The security of this scheme depends only on the secrecy of the factorization of N, and thus we can assume that everyone (including the cryptanalyst) knows N, S, and all the K_i's. Our goal is to prove that the complexity of the root problem remains unchanged even when some of the other roots of the same S (mod N) are given for free. Without loss of generality, it is enough to consider the following pair of problems.

(1) Given N and S, compute R_1.
(2) Given $N, S, R_2, \ldots, R_l$, compute R_1. (The sequence of K_i's and the value of l are assumed to be fixed parameters in these problems).

Since the difficulty of the root problem fluctuates wildly as N goes from 1 to infinity, we would like to establish the equivalence between the security of the RSA cryptosystem and the complexity of our pseudorandom sequences for each value of N rather than asymptotically. To deal with these finite problems, we have to consider their Boolean circuit complexities [6]. Unfortunately, for each particular N, there exists a small circuit that stores the factorization of N and uses it to solve all the root problems mod N efficiently. To overcome this difficulty, we lump together all the moduli N of the same binary size n and claim:

THEOREM 1. *There is a fixed polynomial $P(l, n)$ such that for any number l of known roots, for any size n of the modulus, and for any circuit $C_{l,n}$ that solves all the instances of problem* (2) *of size n, there exists another circuit C'_n of size at most $|C_{l,n}| + P(l, n)$ that solves all the instances of problem* (1) *of size n.*

The peculiar property of the RSA encryption function that makes the proof of this theorem possible is:

LEMMA 2. *There is a polynomial-size circuit that computes from $N, A_1, \ldots, A_l, S^{A_1} (\text{mod } N), \ldots, S^{A_l} (\text{mod } N)$ the value of $S^{A_0} (\text{mod } N)$ where $A_0 = gcd(A_1, \ldots, A_l)$.*

PROOF. By Euclid's algorithm, there are (easy-to-compute) integers $B_1, \ldots, B_l$ such that

$$A_0 = A_1 B_1 + \ldots + A_l B_l.$$

Consequently,

$$S^{A_0} = (S^{A_1})^{B_1} \ldots (S^{A_l})^{B_l} (\text{mod } N),$$

and these exponentiations can be carried out efficiently by the method of repeated squarings. Q.E.D.

COROLLARY. *If the A_i's are relatively prime, then S itself can be computed from its l powers by a circuit of polynomial size.*

PROOF OF THEOREM 1. We show how to construct C'_n from $C_{l,n}$. Given N, S, and all the K_i, we define $T = S^{K_2 \ldots K_l} (\text{mod } N)$. Since $R_1 = S^{1/K_1} (\text{mod } N)$, T is also equal to $R_1^{K_1 K_2 \ldots K_l} (\text{mod } N)$. The following $l - 1$ numbers can be easily computed as powers of S:

$$(2) \qquad T^{1/K_2} = R_1^{K_1 K_3 \ldots K_l} = S^{K_3 \ldots K_l} (\text{mod } N)$$

$$\vdots$$

$$(l) \qquad T^{1/K_l} = R_1^{K_1 K_2 \ldots K_{l-1}} = S^{K_2 \ldots K_{l-1}} (\text{mod } N).$$

The values of N, T, and (2) ... (l) can be fed into $C_{l,n}$ (with T playing the role of the seed S), and the output of this circuit is:

$$(1) \qquad T^{1/K_1} = R_1^{K_2 \ldots K_l} (\text{mod } N).$$

Since the K_i's are pairwise relatively prime, the gcd of the l exponents of R_1 in (1) ... (l) is 1, and thus by the corollary of Lemma 2, we can easily compute R_1 itself. All the computations of powers in (2) ... (l) and the final extraction of R_1 can be done by a circuit whose size is some polynomial in l and n, and thus the size of C'_n does not exceed $|C_{l,n}| + P(l, n)$. Q.E.D.

Practical cryptographic systems must be difficult to break almost everywhere, since the existence of an efficient cryptanalytic algorithm which can decipher one percent of the messages for one percent of the keys is enough to make the system useless. Theorem 1 is not strong enough in the context of cryptocomplexity, since it does not rule out the possibility that the RSA function is secure almost everywhere while our pseudorandom sequence generator is sometimes (i.e., for many S) breakable. To show that this situation is impossible, we have to consider circuits $C_{l,n}$ which are not perfect. For each N of size n, we define $g(N)$ to be the fraction of seeds S for which $C_{l,n}$ computes the correct value of R_1. This success rate depends on the circuit, and its value is typically 1 for easily factorable N. We can now use (for the first time) the randomness of S in order to prove:

THEOREM 2. *There is a fixed polynomial $P(l, n)$ such that for any circuit $C_{l,n}$ that solves some of the instances of problem* (2) *of size n with success rate $g(N)$, there exists another circuit C'_n of size at most $|C_{l,n}| + P(l, n)$ that solves some of the instances of problem* (1) *of size n with success rate at least $g(N)$.*

PROOF. The proof is very similar to the proof of Theorem 1. The new observation we need is that when $K_2, \ldots, K_l$ and N are fixed, the mapping of S values to T values represented by $T = S^{K_2 \ldots K_l} (\text{mod } N)$ is a permutation. Consequently, a randomly chosen S has a probability of $g(N)$ to yield a T for which the oracle $C_{l,n}$ answers correctly. Note that while the *numbers* of easy seeds in

problems (1) and (2) are guaranteed to be similar, their *identities* may be completely different. Q.E.D.

The condition that each K_i must be relatively prime to $\varphi(N)$ is required only in order to guarantee the existence of the appropriate root (e.g., square roots cannot be extracted from seeds S which are quadratic nonresidues mod N). However, if the seeds are chosen in such a way that the roots are known to exist, the proof of Theorem 1 applies to arbitrary K_i's. In particular, it proves that the knowledge of square roots (mod N) cannot possibly aid the cryptanalyst in computing other roots. This is an apparent contradiction to Rabin's result [3] that the ability to extract square roots implies the ability to factor and thus extract arbitrary roots. However, in Rabin's model, S is chosen by the cryptanalyst, whereas in our model, S is chosen by the users of the cryptosystem. Even though S is a randomly chosen number in both models, this seemingly insignificant difference enables the cryptanalyst to factor N in Rabin's model, but leaves him completely in the dark in our model.

The pseudorandom sequence generator we propose is mainly of theoretical interest, since the modular exponentiation of huge numbers is too time-consuming for most practical applications. An interesting open problem is to make the proof technique developed in this paper applicable to faster cryptosystems and one-way functions in order to create more practical sequence generators with guaranteed complexity.

REFERENCES

1. DIFFIE, W. AND HELLMAN, M. Privacy and authentication: An introduction to cryptography. *Proc. IEEE 67*, 3 (March 1979).
2. KAHN, D. *The Code Breakers*, Macmillan, New York, N.Y., 1967.
3. RABIN, M.O. Digitized signatures and public-key functions as intractable as factorization. Tech. Rep. LCS/TR-212, Massachusetts Institute of Technology, Cambridge, Mass., Jan. 1979.
4. RIVEST, R.L. Forwards and backwards encryption. *Cryptologia 4*, 1 (Jan. 1980).
5. RIVEST, R.L., SHAMIR, A., AND ADLEMAN, L.M. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM 21*, 2, (Feb. 1978), 120–126.
6. SAVAGE, J.E. *The Complexity of Computing*. Wiley, New York, N.Y., 1976.
7. SHAMIR, A. On the power of commutativity in cryptography. In *Proc. of ICALP 80, Lecture Notes in Computer Science*, vol. 85, Springer-Verlag, July 1980.

Received February 1982; revised September 1982; accepted September 1982