ON THE GENERATION OF CRYPTOGRAPHICALLY STRONG PSEUDO-RANDOM SEQUENCES

Adi Shamir
Department of Applied Mathematics
The Weizmann Institute of Science
Rehovot, Israel

Abstract

In this paper we show how to generate from a short random seed S a long sequence of pseudo-random numbers R_i in which the problem of computing one more R_i value given an arbitrarily large subset of the other values is provably equivalent to the cryptanalysis of the associated Rivest-Shamir-Adleman encryption function.

I. Introduction

The simplest and safest cryptosystem is undoubtedly the one-time pad, invented by G. S. Vernam in 1917. Its secret key is a long sequence of randomly chosen bits. A cleartext is encrypted by XOR'ing its bits with an initial segment of the key, and the resultant cyphertext is decrypted by XOR'ing its bits again with the same segment. Each segment is deleted after a single use, so that the key is gradually consumed (see Fig. 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.

$$\begin{array}{rl}
\text{cleartexts:} & \overbrace{0\;1\;1\;0}^{\#1}\;\overbrace{0\;1\;0\;0\;0\;1}^{\#2}\;1\;\ldots \\
\text{key:} & 0\;1\;0\;0\;1\;0\;1\;1\;0\;1\;0\;\ldots \\
\hline
\text{cyphertexts:} & \underbrace{0\;0\;1\;0}_{\#1}\;\underbrace{1\;1\;1\;1\;0\;0}_{\#2}\;1\;\ldots
\end{array}$$

Fig. 1

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 practice, this truly random key is replaced by a pseudo-random running key derived during the encryption/decryption process from an initial seed by a sequence generator such as a shift register with non-linear 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 pseudo-random running key must be unpredictable. The main problem is to guarantee that even when the cryptanalyst obtains long segments of the running key (by XOR'ing together known cleartext/cyphertext 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" is notoriously slippery, and it can be defined in any one of the following forms:

(i) The classical notion: the ability to retrieve the desired value from memory.

(ii) The complexity-theoretic notion: The ability to compute the desired value within certain time and space complexity bounds.

(iii) The information-theoretic notion: the ability to sharpen the a-priori probability distribution of candidate values.

The analysis of pseudo-random sequences in this paper is based on definition (ii). Consequently, we do not analyse 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 analyses the information/complexity tradeoffs 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 (ii), 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, and thus a more realistic goal is to develop pseudo-random 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 indentifying 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.

II. Schemes Based on One-Way Functions.

The purpose of this section is to illustrate the trickiness of formal proofs of security by analysing 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 Shamir [1980]).

Given a one-way function f, we can generate a long pseudo-random 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 as 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 encryption function (Rivest, Shamir and Adleman [1978]):

$$E_K(M) = M^K \pmod N \ .$$

This function is believed to be one-way with repsect 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

$$2^K \pmod N \ , \ 3^K \pmod N \ , \ 4^K \pmod N \ , \ \ldots$$

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, etc. 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 $2^K \pmod N$ and (e.g.) the difficulty of computing $5^K \pmod N$ from $2^K \pmod N$ and $3^K \pmod N$.

Another way (proposed by Rivest [1980]) 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 <u>opposite directions</u>:

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

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:

<u>Lemma 1</u>: If f is a one-way function, then a new element of the sequence cannot be computed from a <u>single</u> known element.

<u>Proof</u>: 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 (hard to compute) R 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 f $j-i-1$ 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 pseudo-random 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 diffiuclt. The scheme and its proof are described in the next section.

III. 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 K-th root of the cyphertext $\pmod N$. The cryptographic security of the RSA cryptosystem is thus equivalent by definition to the difficulty of taking roots $\bmod N$. 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 pseudo-random 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.

To actually generate a pseudo-random 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:

$$R_1 = S^{1/K_1} \pmod N \quad , \quad R_2 = S^{1/K_2} \pmod N, \ \ldots \ .$$

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 $S \pmod N$ are given for free. Without loss of generality, it is enough to consider the following pair of problems:

(i) Given N and S, compute R_1.

(ii) Given $N, S, R_2, \ldots R_\ell$, 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 problems 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 pseudo-random sequences for each value of N rather than asymptotically. To deal with these finite problems, we have to consider their boolean circuit complexities. 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 2: There is a fixed polynomial $P(\ell,n)$ such that for any number ℓ of known roots, for any size n of the modulus, and for any circuit $C_{\ell,n}$ that solves all the instances of problem (ii) of size n, there exists another circuit C'_n of size at most $|C_{\ell,n}| + P(\ell,n)$ that solves all the instances of problem (i) of size n.

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

Lemma 3: There is a polynomial size circuit that computes from N, $A_1, \dots, A_\ell$, $S^{A_1} \pmod N, \dots, S^{A_\ell} \pmod N$ the value of $S^{A_o} \pmod N$ where $A_o = \gcd(A_1, \dots, A_\ell)$.

Proof: By Euclid's algorithm, there are (easy to compute) integers $B_1, \dots, B_\ell$ such that

$$A_o = A_1B_1 + \dots + A_\ell B_\ell \ .$$

Consequently,

$$S^{A_o} = \left(S^{A_1}\right)^{B_1} \cdots \left(S^{A_\ell}\right)^{B_\ell} \pmod N \ ,$$

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

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 2: We show how to construct C'_n from $C_{\ell,n}$. Given N,S and all the K_i, we define $T = S^{K_2 \cdots K_\ell} \pmod N$. Since $R_1 = S^{1/K_1} \pmod N$, T is also equal to $R_1^{K_1K_2 \cdots K_\ell} \pmod N$. The following $\ell-1$ numbers can be easily computed as powers of S:

$$(2) \quad T^{1/K_2} = R_1^{K_1K_3 \cdots K_\ell} = S^{K_3 \cdots K_\ell} \pmod N$$

$$\vdots$$

$$(\ell) \quad T^{1/K_\ell} = R_1^{K_1K_2 \cdots K_{\ell-1}} = S^{K_2 \cdots K_{\ell-1}} \pmod N \ .$$

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

$$(1) \quad T^{1/K_1} = R_1^{K_2 \cdots K_\ell} \pmod N \ .$$

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 3 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 $|C_{\ell,n}| + P(\ell,n)$.

Q.E.D.

Practical cryptographic systems must be almost everywhere difficult to break, since the existence of an efficient cryptanalytic algorithm which for one percent of the keys can decypher one percent of the messages is enough to make the system useless. Theorem 2 is not strong enough in the context of cryptocomplexity, since it does not rule out the possibility that the RSA function is almost everywhere secure while our pseudo-random sequence generator is sometimes (i.e., for many S) breakable. To show that this situation is impossible, we have to consider circuits $C_{\ell,n}$ which are not perfect. For each N of size n, we define $g(N)$ to be the fraction of seeds S for which $C_{\ell,n}$ 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 4: There is a fixed polynomial $P(\ell,n)$ such that for any circuit $C_{\ell,n}$ that solves some of the instances of problem (ii) of size n with success rate $g(N)$, there exists another circuit C_n' of size at most $|C_{\ell,n}| + P(\ell,n)$ that solves some of the instances of problem (i) of size n with success rate at least $g(N)$.

Proof: The proof is very similar to the proof of Theorem 2. The new observation we need is that when $K_2, \ldots K_\ell$ and N are fixed, the mapping of S values to T values represented by $T = S^{K_2 \cdots K_\ell} \pmod N$ is a permutation. Consequently, a randomly chosen S has a probability of $g(N)$ to yield a T for which the oracle $C_{\ell,n}$ answers correctly. Note that while the numbers of easy seeds in problems (i) and (ii) are guaranteed to be similar, their identities may be completely different.

Q.E.D.

The pseudo-random 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.

Bibliography

1. Rivest [1980] - private communication.
2. Rivest, Shamir and Adleman [1978] - "A method for obtaining digital signatures and Public-Key Cryptosystems", CACM, Vol. 21, No. 2, February 1978.
3. Shamir [1980] - "On the Power of Commutativity in Cryptography", ICALP Proceedings, July 1980.