

INFERRING A SEQUENCE GENERATED BY A LINEAR CONGRUENCE

Joan B. Plumstead

Computer Science Division
University of California
Berkeley, California

ABSTRACT

A pseudo-random number generator is considered cryptographically secure if, even when a cryptanalyst has obtained long segments of the generator's output, he or she is unable to compute any other segment within certain time and space complexity bounds. A pseudo-random number generator which is as cryptographically secure as the Rivest-Shamir-Adleman encryption scheme is presented in [Shamir]. This method for generating pseudo-random numbers is quite slow, though, and it is not known whether any statistical biases might be present in the sequences it generates. Blum and Micali [BlMi] give a pseudo-random bit generator, with arbitrarily small bias, which is cryptographically strong, assuming the problem of index finding is intractable. But their method is also slow. Other cryptographically strong, but slow, pseudo-random bit generators are given in [BBS] and [Yao]. This suggests the question of whether any of the pseudo-random number generators commonly in use are also cryptographically secure. In particular, the linear congruential method, $X_{i+1} = aX_i + b \pmod{m}$, is very popular and fast. Obviously, this method is not cryptographically secure if the modulus, m , is known. In that case, one could solve for x in the congruence $(X_2 - X_1) \equiv x(X_1 - X_0) \pmod{m}$. Then the remainder of the sequence could be correctly predicted using $X_{i+1} = x(X_i) + (X_1 - x(X_0)) \pmod{m}$. In [K1980], Knuth has discussed this problem, assuming m is known and is a power of two, but assuming that only the high order bits of the numbers generated are actually used. We have looked at the problem, assuming the m is unknown and arbitrary, but that the low order bits are also used. We have shown that, under these assumptions, the linear congruential

method is cryptographically insecure. A similar result is given in [Reeds], but, among other problems, that result relies on the assumption that factoring is easy.

To reiterate, assume that a fixed linear congruential random number generator, $X_{i+1} = aX_i + b \pmod{m}$, is given, but the constants a , b , and m are unknown. The problem is to predict, the remainder of the sequence from some of the X_i . First we consider the problem of finding an \hat{a} and a \hat{b} such that $X_{i+1} = \hat{a}X_i + \hat{b} \pmod{m}$ for all $i \geq 0$, without finding m . The two algorithms we have found to do this use only an initial segment of the sequence. The first algorithm is somewhat easier, but the second is optimal in the sense that it never requires knowledge of more of the X_i than are necessary to completely determine \hat{a} . In the worst case, neither algorithm needs an initial segment of length greater than $2 + \lceil \log_2 m \rceil$.

Then, we use a modified version of the second of these algorithms to predict m and the unknown portion of the sequence. Here the problem becomes one of inference. In some cases, it may take a long time to find m , even though many correct predictions for the X_i can be made before m can be uniquely determined. The algorithm we found looks at X_0 , X_1 , and X_2 , and then begins predicting the X_i . We assume that whenever an incorrect prediction occurs, the correct value is revealed before the next prediction is made. In a few cases, X_0 , X_1 , and X_2 will be sufficient to determine an \hat{a} and a \hat{b} such that $X_{i+1} = \hat{a}X_i + \hat{b} \pmod{m}$ for all $i \geq 0$. Otherwise, when the first incorrect prediction is made, suitable \hat{a} and \hat{b} are determined from the correct X_i value, as well as an initial guess for m . When further errors are made in predicting the X_i , m is updated so that it is consistent with all previous data. Analysis of the algorithm shows that no more than $2 + \log_2 m$ updates need ever be made.

Finally, we show that, if some of the X_i with $i > j$ are known, they can also be useful in predicting X_j .

The algorithms described above and the proofs of these results can be found in [Pl].

REFERENCES

- [BBS] Blum, L., Blum, M., and Shub, M., *A Simple Secure Pseudo-Random Number Generator*, Advances in Cryptography: Proceedings of CRYPTO 82, 1982.
- [BlMi] Blum, M., and Micali, S., *How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits*, Proc. 23rd IEEE Symp. on Foundations of Computer Science, 1982.

- [K1980] Knuth, D.E., *Deciphering a Linear Congruential Encryption*, Technical Report 024800, Stanford University, 1980.
- [P1] Plumstead, J., *Inferring a Sequence Generated by a Linear Congruence*, Proc. 23rd IEEE Symp. on Foundations of Computer Science, 1982.
- [Reeds] Reeds, J. "Cracking" a Random Number Generator, *Cryptologia*, Vol. 1, January 1977.
- [Shamir] Shamir, A., *On the Generation of Cryptographically Strong Pseudo-Random Sequences*, International Colloquium on Automata, Languages, and Programming, 7th, 1980.
- [Yao] Yao, A., *Theory and Applications of Trapdoor Functions*, Proc. 23rd IEEE Symp. on Foundations of Computer Science, 1982.

