The Shrinking Generator: some practical considerations

Hugo Krawczyk

IBM T.J. Watson Research Center
Yorktown Heights, NY 10598

(Abstract)

The Shrinking Generator, presented at Crypto'93, is a LFSR-based pseudo-random generator suitable for the implementation of additive stream ciphers. It is particularly simple and has attractive security properties. (The reader is referred to [1] for the definition of the generator and its properties). Although the algorithm was originally intended for hardware implementation, here we will focus on some initial results of an experimental software implementation and other practical considerations.

In particular, we present the advantages of using variable and secret connections for the linear feedback shift registers as opposed to using fixed and known connections. Not only do they enhance security significantly, they also provide more flexibility to the resultant cryptosystem (like scalable security, connection refreshment, etc.). We also discuss some of the efficient methods for choosing these connections, namely, choosing random primitive polynomials over GF(2). Efficient algorithms for this task have a complexity of $O(n^2)$ word operations (mostly, shifts and exclusive-or). Reference [3] contains a very simple algorithm (see also [4, 5]).

We present the results of an experimental software implementation of the shrinking generator carried out by Erol Basturk in IBM. The algorithm was programmed to support random variable connections. In particular, this means that no assumption on sparse connections or heavy pre-processing for a particular connection are done. The resultant implementation (coded in C) achieves a throughput of 2.5 Mbit/second on an IBM workstation running at 33MHz. These results correspond to both register lengths in the range of 61-64 bits each. The *currently known* effective key length for such a construction is of about 110 bits (about twice the length of register S since we are using secret random connections). The implementation of the LFSRs is done by exploiting the fact that each block of LFSR output (say 32 bits) can be computed as a linear transformation, i.e. matrix multiplication, of the previous block of output (see, e.g., [2]). The matrix involved depends only on the connections of the corresponding LFSR, and the matrix multiplication operation is speeded up through a table look-up mechanism. Finally, the shrinking operation (between the two sequences) is also performed using a table-lookup mechanism.

We stress that for slower applications a more straightforward implementation can be used. On the other hand, implementations faster than ours can be achieved if one uses sparse connections. But then the penalty in security can be significant. In such a case, which we do not recommend, the registers have to be chosen to be considerably longer. As a final remark we stress that although most of the known attacks on the shrinking generator depend primarily on the length of the selecting register S, we recommend the other register, A, to be of comparable length as S.

Acknowledgment

We are grateful to Erol Baştürk for his software implementation of the shrinking generator, and to Bill Chambers for pointing out to us reference [3].

References

1. Coppersmith D., Krawczyk, H., and Mansour Y., "The Shrinking Generator", presented at Crypto'93.
2. S.W. Golomb, *Shift Register Sequences*, Aegean Park Press, 1982.
3. John A Gordon, "Very simple method to find the minimal polynomial of an arbitrary non-zero element of a finite field", *Electronics Letters* Vol. 12, 1976, pp. 663-664.
4. Rabin, M.O., "Probabilistic Algorithms in Finite Fields", *SIAM J. on Computing*, Vol. 9, 1980, pp. 273-280.
5. Rabin, M.O., "Fingerprinting by Random Polynomials", Tech. Rep. TR-15-81, Center for Research in Computing Technology, Harvard Univ., Cambridge, Mass., 1981.