TWO STREAM CIPHERS

W. G. Chambers

Department of Electronic and Electrical Engineering,
King's College London, Strand, London WC2R 2LS, UK

1 Introduction

Two contrasting keystream generators are considered. The first uses a shift register of length n operating with arithmetic carried out modulo 2^e. The feedback has been made non-linear by using the bit-by-bit exclusive-or function as well as the linear operation of addition. The second generator is a cascade of clock-controlled shift-registers with several bits passed from stage to stage, through invertible scramblers or "S-boxes". The first generator uses a large number of multiplications and is intended for use on digital signal processors.

These generators have some points of resemblance: (a) They can both be regarded as cascades of binary shift registers with primitive feedbacks; (b) there is some theory available for the periods (and for the linear equivalences); (c) they can have multi-bit outputs; (d) the key-space can be very large.

2 A Modified Linear Congruential Generator

2.1 Description

We start by considering linear recursions of the form

$$a_{t+n} = \sum_{j=0}^{n-1} c_j a_{t+j} \bmod 2^e \text{ for } t = 0, 1, 2, \ldots \tag{1}$$

with $a_0, a_1, \ldots, a_{n-1}$ specifying the initial conditions. At least one of these values is odd. Here a_t and c_j are e-bit integers. We may then derive a binary output by picking the most significant bit of each a_t, and an integer output by picking the k most significant bits ($k \leq e$). It should be a particularly convenient way of generating pseudo-random sequences technique on some digital signal processors which have high-speed facilities for multiply-accumulation. Now as regards the cryptologic security: The generator is a linear congruential generator and crypt-analytic techniques are available at least when the coefficients are known [1]. To increase the security we propose the use of the bit-by-bit exclusive-or function as a source of non-linearity inside the recursion; the exclusive-or function is a fast operation on most microprocessors and digital signal processors. To make the discussion definite we consider a recursion of the form

$$a_t = \left(\sum_{j=0}^{n-1} c_j a_{t+j} \bmod 2^e \right) \text{ XOR } \left(\sum_{j=0}^{n-1} d_j a_{t+j} \bmod 2^e \right) \text{ for } t = 0, 1, 2, \ldots \tag{2}$$

where XOR denotes a bit-by-bit exclusive-or. The *base polynomial* $h(x) = x^n + \sum_{j=0}^{n-1}(c_j + d_j)x^j \bmod 2$ is a primitive binary polynomial.

The period of the generator (1) has been investigated long ago in [2]; more recent work concerns the period and upper and lower bounds on the linear equivalence of the binary sequences produced by taking the bit of a given order of significance from each a_t. (See [4] for references.) Let a_t have the binary decomposition $a_t = \sum_{i=0}^{e-1} a_{t,i}2^i$ with $a_{t,i} \in \{0,1\}$. Denote the sequence $\{a_0, a_1, a_2, \ldots\}$ by α and the binary sequences $\{a_{0,i}, a_{1,i}, a_{2,i}, \ldots\}$ by α_i. We quote the following results: If the base polynomial $h(x)$ is a primitive binary polynomial of degree n, the possible periods of α_i are $2^k(2^n - 1)$ with $k = 0, 1, \ldots, i$. Moreover for any i satisfying $1 < i < e$ and with $h(x) = x^n + f(x) \bmod 2$ a specified primitive polynomial and with α_0 not identically zero, all but a fraction $2/2^n$ of the possible connection polynomials $f(x) = \sum_{i=0}^{n-1} c_i x^i$ give α_i the maximal period $(2^n - 1)2^i$. From the practical point of view this means that provided we keep n reasonably large, say > 40, there is not much risk of obtaining a short-period sequence. There are also "fast" tests for checking that the period is maximal [3].

These results were derived using the linearity of (1). What can be said about the periods of sequences generated by (2) depends very much on the d_j. If any of the d_j are odd we can say very little apart from the fact that the period is a factor of $(2^n - 1)2^{e-1}$. Much more definite conclusions apply if all the d_j are even. (There may be a price to pay for this increased understanding in that the generator may not be quite as strong cryptologically as in the general case.) Then the occurrence of short periods depends only on the feedback coefficients taken mod 4, and a fast test is given in [4]. With the above provisos the probability that α_k has a short period for $k \geq 3$ is $4/2^n - 4/4^n$.

2.2 Implementation

Reasonably large values of n (~ 100 say) are needed for long periods, but to reduce the number of terms in (2) we need c_i and d_i to be non-zero only for $i \leq l$ where l is an integer considerably less than n. In particular the base polynomial $h(x)$ must be of the form $x^n + g(x)$ with $\deg g(x) \leq l$. Random or key-dependent choices of $h(x)$ of a special form must then be tested for primitiveness as described for instance in [8]. However when n is a Mersenne exponent, making $N = 2^n - 1$ a prime, we simply have to check that $x^{2^n} \equiv x \bmod (h(x), 2)$; this is done by n modular squaring operations (linear in GF(2)). (Examples of Mersenne exponents are 31, 61, 89, 107 and 127.) When n is a Mersenne exponent the probability that a randomly chosen polynomial of degree n is primitive is approximately $1/n$, so the search is not extensive, especially as multiples of x and of $x + 1$ are speedily eliminated.

2.3 Further Remarks

The overall system may be regarded as a cascade of binary linear feedback shift-registers, one shift-register for each order of significance, with non-linear feed-

forward from shift-registers of a lower order of significance to those of higher. It should be noted that the XOR coupling is strictly feed-forward in this sense.

The worry about this kind of generator is whether it is a little too amenable to mathematical analysis, so that cryptologic weaknesses can be found along the lines [1] for the generator without the XOR function. The periods are short, being of order 2^{n+e}, whereas the number of states is of order 2^{ne}, and this may be another cause for concern. The resemblance of this generator to the usual linear congruential generators used for random-number generation suggests that it should not fail the conventional tests for randomness.

3 A Clock-controlled Cascade

3.1 Introduction

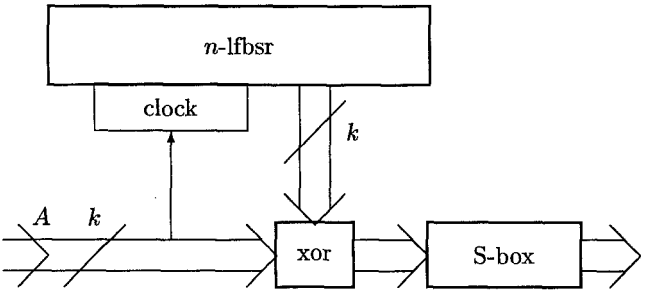

Fig. 1. A stage of the clock-controlled cascade

A clock-controlled cascade [5] is built up with a number of stages each as shown in Fig. 1. The input A on the left consists of a sequence of k-bit integers. One or more of these bits is used to control the stepping of a linear feedback shift-register (lfsbr) of size n ($n \geq k$) with a primitive feedback polynomial and with a non-zero initial setting. (The value of n is the same in every stage, although the feedback can vary.) The input is then XOR'd with k bits from the lfbsr and passed through an invertible S-box which gives a reversible one-to-one onto mapping of k-bit patterns to k-bit patterns. The initial stage of the cascade is clocked regularly. We shall henceforth choose $k = n$. The main results for a cascade of K stages are that the period is almost maximal (equal to $(2^n - 1)^K$) and that the k-bit outputs are distributed as uniformly as possible over a period [5].

We now discuss three aspects of this system, the lfbsrs, the S-box, and the clocking.

3.2 The Shift Registers

The obvious choice for the length n of each lfbsr is 32, the word-length on most modern microprocessors. The primitive feedback polynomials should be chosen "at random", that is under the control of the key. Since the feedback polynomials are not required to have a special form we can select a fixed primitive element α in GF(2^n), specified as the root of a primitive polynomial, and then using a technique such as that in [6] we find the minimal polynomial of α^s where s is relatively prime to $N = 2^n - 1$ [7]. Such an s is found "at random" as follows: Let $p_1, p_2, \ldots, p_l$ denote the prime factors of N. We choose "at random" l integers s_i satisfying $1 \leq s_i < p_i$ and then set $s = \sum_i (N/p_i)s_i \bmod N$. In the case $n = 32$ there are five prime factors $2 + 1$, $2^2 + 1$, $2^4 + 1$, $2^8 + 1$, and $2^{16} + 1$, so that the s_i can be specified by 31 bits in all. (Things are simpler for $n = 31$, for then $N = 2^n - 1$ is a prime, and the minimal polynomial of any element in GF(2^{31}) apart from 0 and 1 is primitive.)

3.3 An Invertible S-box

A 32-bit invertible S-box can be set up using a technique suggested by David Wheeler at this workshop. Let t be a 256-element table of 32-bit words with the least significant 24 bits in each word chosen "at random", and the most significant 8-bit bytes forming a "random" permutation of 0 to 255. Then we may set

$$\text{Sbox}(x) = (x >> 8) \text{ XOR } t[x \text{ AND } 255]$$

Here $>>$ denotes a logical right-shift, with zero-fill on the left. This not only gives a Feistel-like invertible mapping, but also in effect provides an 8-bit right-circular shift.

3.4 The Clocking

The simplest clocking technique is to use a single bit from the input A to control whether the step should be 1 or 2 [5]. However with a little care a more elaborate arrangement can be used, giving greater diffusion. To maintain the periodicity of the cascade it is necessary to ensure that the number of steps S taken by the clocked lfbsr over an input cycle of period N^r (after r stages) is relatively prime to N [5]. (Here $N = 2^n - 1$.) Suppose we use 2 bits from A to determine whether the number of steps should be a, b, c or d. The 4 possible values of the bit-pair will have frequencies $(N^r - (-1)^r)/4$, $(N^r - (-1)^r)/4$, $(N^r - (-1)^r)/4$, and $(N^r + 3(-1)^r)/4$ in some order [5]. (Note that $N^r \equiv (-1)^r \bmod 4$, and that the distribution is the most uniform possible.) Let the corresponding numbers of steps be x, y, z, t, a permutation of a, b, c, d. Then we find $4S \bmod N = (-1)^r(3t - x - y - z)$. In the case when $(a, b, c, d) = (1, 2, 3, 4)$ we find that this has four possibilities $\pm 6, \pm 2$, and so cannot be used for $n = 32$ since $N = 2^{32} - 1$ is divisible by 3, and may not be relatively prime to S. However the choice $(a, b, c, d) = (1, 2, 4, 5)$ solves this problem, the possible values for $4S \bmod N$ then being ± 8, ± 4.

The stepping can be speeded up by using a table giving the overall feedback after a multiple step. Thus for a 5-fold step we use a table with 32 entries. The least significant 5 bits in the lfbsr are used to select an entry from the table which is XOR'd with the lfbsr after the lfbsr has been logically right-shifted by 5 places.

3.5 Use of a Cycling Counter

The n-lfbsr can be replaced by a counter cycling through the values $1, \ldots, 2^n - 1$. The clock-control is driven by q bits ($q \leq k$) selecting a value $J[a]$ at the address a ($0 \leq a \leq Q$) in a table of unsigned n-bit integers. Here Q equals $2^q - 1$. The counter is then stepped from its old value u to a value v obtained as follows: First set $v = u + J[a] \bmod 2^n$, and if $v < u$ increment v by 1. Over an input cycle of period N^r we find that all but one of the 2^q possibilities for a occur with frequency $(N^r - (-1)^r)/2^q$; the exceptional value α occurs $(-1)^r + (N^r - (-1)^r)/2^q$ times. Thus the number of steps S taken by the counter over an input cycle is $((N^r - (-1)^r)/2^q)(\sum_{a=0}^{Q} J[a]) + (-1)^r J[\alpha]$. This must be relatively prime to N, so that $2^q S \bmod N = (-1)^r((\sum_{a=0}^{Q} J[a]) - 2^q J[\alpha])$ must be relatively prime to N. We do not know which is the special value α, so we choose all the $J[a]$ relatively prime to N, and also arrange that $\sum_{a=0}^{Q} J[a] \equiv 0 \bmod N$. This is done in the same way as the exponent s was found in Sec. 3.2. For $a = 0$ to $Q - 1$ we choose "random" integers s_{ia} with $1 \leq s_{ia} < p_i$ and set $J[a] = \sum_i (N/p_i) s_{ia} \bmod N$. Then we have $\sum_{a=0}^{Q-1} J[a] \equiv \sum_i (N/p_i) x_i \bmod N$ with $x_i = (\sum_{a=0}^{Q-1} s_{ia}) \bmod p_i$. If any of the x_i vanish we choose another non-zero value of $s_{i,Q-1}$, less than p_i; finally we set $s_{iQ} = p_i - x_i$ to find $J[Q]$.

References

1. A. M. Frieze, J. Hastad, R. Kannan, J. C. Lagarias, A. Shamir, "Reconstructing truncated integer variables satisfying linear congruences", *SIAM J. Comput.*, **17**, 262-280 (1988)
2. M. Ward, "The arithmetical theory of linear recurring series", *Transactions of the American Mathematical Society*, **35**, 600-628 (July 1933)
3. A. D. Barnard, J. R. Silvester, W. G. Chambers, "Guaranteeing the period of linear recurring sequences (mod 2^e)", *IEE Proceedings-E*, **140**. 243-245, (Sept 1993)
4. W. G. Chambers, Z-D. Dai, "On binary sequences from recursions 'modulo 2^e' made non-linear by the bit-by-bit 'XOR' function", *Lecture Notes in Computer Science*, **547**, 200-204, 1991
5. W. G. Chambers, "Clock-controlled shift registers in binary sequence generators", *IEE Proceedings E*, **135**, 17-24 (1988)
6. J. A. Gordon, "Very simple method to find the minimal polynomial of an arbitrary non-zero element of a finite field", *Electronics Letters* **12**, 663-664 (Dec 1976)
7. E. R. Berlekamp, *Algebraic Coding Theory*, (New York: McGraw-Hill) 1968; Section 4.2
8. B. J. M Smeets, W. G. Chambers, "Windmill *pn*-sequence generators", *IEE Proceedings E*, **136**, 401-404 (Sept 1989)