Annals of University of Craiova, Math. Comp. Sci. Ser.
Volume 36(2), 2009, Pages 42–46
ISSN: 1223-6934

Combining Linear Feedback Shift Registers

Nicolae Constantinescu

Abstract. In cryptography, LFSRs are often used mainly because they are extremely fast and easy to implement. LFSRs do not offer a high security, but there are some generators that use a combination of several LFSRs outputs in order to obtain a more secure keystream[6]. In this paper is presented such a generator which is based on the complexity of the LFSRs used. This generator can be used with a cipher system which has a simetric key encryption. The security of such a cipher will be much higher[8].

2000 Mathematics Subject Classification. Primary 94A55; Secondary 11T71, 68P25.
Key words and phrases. LFSR. Berlekamp Massey Cryptographyc Attack.

1. Introduction

The evaluation problem for a cipher system is finding the output data with the minimal size from which we can find some information about the algorithm, the key or the plaintext. On the other hand, cracking the system means finding the output data with the maximum size from which we cand find the plaintext ([9] and [10]).

Almost all cyptographic systems use pseudorandom numbers. These numbers are chosen with a pseudorandom number generator which uses a recursive aplication. A RNG starts with an arbitrary state, using a seed state. The RNG should always produce the same result if the same state is used. The maximum size of the sequence before it begins to repeat is given by the size of the state. So, if the state of a RNG has n bits, its period cannot be longer than 2^n results.

The use of pseudorandom numbers is preffered because they are efficient to generate and, unlike secure numbers, the pseudorandom numbers are independent of each other. This means that if someone discovers one of the numbers, he cannot break the other values using the discovered one. For cryptographic RNGs, this is not true because the values are not independent. Another advantage of using the RNG is the fact that the starting state may not be secret. The only important thing is that the RNG's output must be collision resistant.[4].

Definition 1.1. *(Vaduva* [7]*). A RNG (pseudorandom number generator) is an algorithm which generates a sequance of numbers that approximates the properties of random numbers. It has the structure:* $G = (S, \mu, f, U, g)$ *where* S *is a finit lots of states;* μ *is a distribution of probability on* S *named initialy distribution;* $f : S \to S$ *is a transition function;* U *is lots of output symbols;* $g : S \to U$ *is an output function.*

The use of a suitable RNG cannot be distinguish from the use of a random sequence. The security of the cryptographic systems which use RNGs is based on this fact. The simplest example of this are the stream ciphers. They use the addition modulo 2 between the plaintext and the RNG. The result is the ciphertext ([5] and [4]):

$$c_i = m_i \oplus g(k_i)$$

where c_i is the ciphertext, k_i is the i^{th} key derived from the base key k, g is the RNG. We can write this using the encyrption function:

$$f(m_i, k_i) = m_i \oplus g(k_i)$$

where the encryption scheme is called stream encryption.

If we know the encryption function $f(.;.)$ we can find the message m_i from c_i:

$$m_i = h(c_i, k_i)$$

2. LFSR and Linear Complexity

A LFSR (linear feedback shift register) is a machine with m registers $R_0, \dots R_{m-1}$ and a XOR gate. Every register holds a single bit which can be or cannot be connected to the XOR gate.

Definition 2.1. *A LFSR has m registers and a feedback function. If $a(t)$ is a new element then:*

$$a(t) = g(a(t-1), \dots, a(t-n+1)) \oplus a(t-n)$$

where $\oplus$ is the addition modulo 2 and g is the output function of the generator.

A generator based on LFSRs is the Gegge Generator which combines three LFSRs. The combination is non-liniar:

$LFSR_2$ and $LFSR_3$ give the input of the generator and $LFSR_1$ is the selection function. If a_1, a_2, a_3 are the outputs of the three LFSRs used then the output of the Geffe generator is given by [3]:

$$b = (a_1 \wedge a_2) \oplus (\overline{a_1} \wedge a_3)$$

The generator's period is the least common multiple of the three LFSRs. The Geffe generator can be extended to $2^k + 1$ LFSRs, but this will not encrease the security of the generator.

As we know, the LFSRs do not offer a high security. To increase the lever of security the LFSR must have a higher linear complexity. The linear complexity of the sequance S is the size of the smallest LFSR which generates this sequance. If such a LFSR does not exist then we have an infinite linear complexity. For example, if we have a sequence $\{S_i\}$ of length n we know that there exists a LFSR with n registers that generates this sequence, so its linear complexity is at most n. However, we will, often, have another LFSR with fewer registers which will also generates the same sequence. To compute the linear complexity we will present the Berlekamp Massey algorithm.

2.1. Berlekamp Massey algorithm. Mathematical Backround. This algorithm finds the shortest LFSR for a given p-ary sequance. In other words, its result is the minimal polynomial of a lineary recurrent sequence. For $p = 2$ we have a binary sequence. Our motivation for considering the linear complexity of binary sequences is that if we use some complicated method for generating a binary sequence to use as a keystream it would be somewhat disconcerting to discover that in fact the sequence could be generated by a single LFSR with comparatively few registers. Hence we should be careful to use sequences with high linear complexity [2]. Next we will describe the stepts of this algorithm:

(1) $s_0 s_1 \dots s_n$ is the given sequence;
(2) we have three arrays $b_0 \dots b_n$, $c_0 \dots c_n$ and $t_0 \dots t_n$ and $b_0 = 1$, $c_0 = 1$, $L = 0$, $N = 0$, $m = -1$, $p = 2$ where m is the largest integer smaller than N such that $L(s^m) < L(S^N)$;
(3) compute $d = (s_N + c_1 s_{N-1} + c_2 s_{N-2} + \dots + c_L s_{N-L}) \ mod \ p$;
(4) if $d = 0$ then c is already a polinomyal which annihilates the stream portion from $N - L$ to N and $L(s^{N+1}) = L$;
(5) if $d \neq 0$ then:
    (a) t is a copy of c;
    (b) $c_{N-m} = c_{N-m} \oplus b_0$, $c_{N-m+1} = c_{N-m+1} \oplus b_1 \dots c_{n-1} = c_{n-1} \oplus b_{n-N+m-1}$ where $\oplus$ is addition modulo 2.
    (c) if $L \leq \frac{N}{2}$ then $L = N + 1 - L$ $m = L$ and $b = t$.

2.2. Berlekamp-Massey Implementation. The Berlekamp Massey Algorithm has a running time $O(n^2)$ [2]. The implementation is described below:

Require: $s_0 s_1 \dots s_n$
Ensure: $L(s)$
Start
$b_0 \leftarrow 1$
$c_0 \leftarrow 1$
$N \leftarrow 0$
$L \leftarrow 0$
$m \leftarrow -1$
$p \leftarrow 2$
while N¡n **do**
    $d \leftarrow (s_N + c_1 s_{N-1} + c_2 s_{N-2} + \dots + c_L s_{N-L}) \ mod \ p$
end while
if $d = 1$ **then**
    $t \leftarrow c$
    for $j = N - m$ to $n - 1$ **do**
        $c_j \leftarrow c_j \oplus b_i$
        $i = i + 1$
    end for
    if $L \leq \frac{N}{2}$ **then**
        $L \leftarrow N + 1 - L$
        $m \leftarrow N$
        $b \leftarrow t$
    end if
    $N = N + 1$
end if
Stop

3. Algorithm

In this work it is proposed a new type of Gollman generators (based on many shift waterfalling registers, like [15, 16, 17]). The algorithm is based on a LFSR with the registers R_i. The algorithm takes the output of the registers and applies a boolean function. All the registers, except the first (R_0) which has a constant tact, are modified by the value of the tact cell of the previous register. These changes are also made

applying the function F_i.

The parameters used in the algorithm are:
- n is the number of the removed registers;
- $deg\ f_i, 0 \leq i \leq n-1$ is the degree of the feedback polinom;
- k is the size of the cell;
- p is the maximum number of rotations (p is minimum 2).
- F_i is the tact function defined as:

$$F_i = \begin{cases} 1 & if\ MSB[tact_{i-1}] = 0 \\ 2 & \end{cases}$$

- the last register has the tact given by:

$$F_i = \begin{cases} 1 & if\ MSB[tact_{n-2} \oplus MSB[R_{n-1}[deg-1]]] = 0 \\ 2 & \end{cases}$$

- out_0 is the cell of the first register's output;

$$y_{n-1} = R_{n-1}[R_{n-2}[out_{n-1}] + R_{n-1}[deg+1] >> (k + \log_2(deg))\ mod\ deg]$$

and $R_{n-1}[deg-1] >> (k - \log_2(deg))$ are the first seven bits of the cell $deg-1$ of the last register.

The randomize is described below:

- the tact cell for the next register R_i is $tact_{i-1} \in \{0, \ldots, deg-1\}$ for $0 \leq i \leq n-1$

We will now describe the algorithm which generates k bits:

For rotating the first register R_0 we have to replace the output of this register $y_0 = R_0[out_0]$ with the value from the cell $tact_0$ resulting $R_0[tact_0]$.

$$F_i(tact_0) \in \{1, \ldots, p\}$$

1: Start
2: **for** i=0 to n-1 **do**
3: rotate R_i by $F_i(tact_i)$ times
4: $y_i = R_i[R_{i-1}[out_i]\ mod\ deg]$
5: compute the next tact $F_i(tact_i) \in \{1, \ldots, p\}$
6: **end for**
7: output $y = f(y_1 \ldots y_n)$
8: Stop

In line 4 the output y_i of the register R_i is taken from the adress:

$$R_{i-1}[out_{i-1}]\ mod\ deg$$

The feedback relations of the parametric functions are computed in modulo 2^k.
The function $f(y_1 \ldots y_n)$ has the value given by $y_1 \oplus \ldots y_n$.
The tact functions are given by the formula:

$$F_i = \begin{cases} 1 & if\ LSB[tact_{i-1}] = 0 \forall i = 1, \ldots, n \\ 2 & \end{cases}$$

4. Conclusions

The problem of the pseudorandom number generation is real important because there is a close relashionship between encryption and randomness. As we know, the security of the encryption algorithms usually depends on the random choice of keys and bit sequence. Ciphers with perfect secrecy require randomly chosen key strings that have the same length as the encrypted message [1]. The most popular generators are the ones based on LFSRs mostly because they are efficient and easy to implement. The most used method for testing the security is the attack based on the Berlekamp Massey Algorithm (Maurer [11], Menicocci [10], Meyerm [13], Golic [14]).The future research is based on the Berlekamp-Massey attack study, which conclude to optimize the register length and initial values from them related to the attack complexity.

References

[1] Hans Delfs and Helmut Knebl,Introduction to Cryptography. Principles and Applications, *Springer 2007*.
[2] JOHN TALBOT, Complexity and Cryptography An Introduction, *Cambridge 2006*.
[3] P. L Ecuyer, *Random Numbers for Simulation*, Comm ACM 33, 10(1990), 742-749, 774
[4] J. Viega, Practical Random Number Generation in Software, in *Proc. 19th Annual Computer Security Applications Conference*, Dec. 2003,1-4.
[5] Michael Luby, Pseudorandomness and Cryptographic Applications, *Princeton Univ Press*, 1996.
[6] RICHARD A. MOLLIN, An INTRODUCTION to CRYPTOGRAPHY, *CRC 2007*.
[7] Ion Vaduva, Modele de simulare cu Calculatorul, *Ed. Tehnica*, 1977.
[8] K.C. Zeng, C.H. Yang, T.R.N. Rao, An improved linear syndrome algorithm in cryptanalysis with application, Proceedings Crypto'89, *Springer-Verlag Lecture Notes in Computer Science*, **435**, 1989.
[9] J.L. Massey, Shift-register synthesis and BCH decoding, *IEEE Trans. Information Theory*, IT-15, (1), 122-127.
[10] J.L. Massey, R.A. Rueppel, Linear ciphers and random sequence generators with multiple clocks, Proceedings Eurocrypt'84, *Springer-Verlag Lecture Notes in Computer Science*, 209, 74-87, 1984.
[11] U. Maurer, Cascade Chipers:The importance of being first, *J. of Cryptology*, 6, 1993.
[12] R. Menicocci, A systematic Attack on clocked controlled cascades, *Adv. in Cryptology-Eurocrypt*, 1994.
[13] W. Meyerm, O. Staffelback, Fast corellation attacks on certain stream ciphers, *Journal of Cryptology*, 1, 159-176, 1989.
[14] J. Golic, Cryptanalysis of the Alleged A5 Stream Cipher, *Adv. in Cryptology-Eurocrypt*, 1997.
[15] C. Jansen, D. Boekee, The shortest feedback shift register that can generate a given sequence, *Adv. in Cryptology -Crypto*, 90-99, 1990.
[16] B. Arazi, On the synthesis of De Bruijn sequences, *Information and Control*, 49,(**2**), 81-90, 1981.
[17] A. Lempel, On a homomorphism of De Bruijn Graph and its application to the design of Feedback shift registers, *IEEE Transactions on Computers*, C-19, (**12**), 1204-1209, 1970.

(Nicolae Constantinescu) Department of Computer Sciences, University of Craiova, Al.I. Cuza Street, No. 13, Craiova RO-200585, Romania, Tel. & Fax: 40-251412673
E-mail address: nikyc@central.ucv.ro