Buletinul Ştiinţific al Universităţii Politehnica" din Timişoara
Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 50(64), Fascicola 2, 2005

A comparison between a linear feed-back shift register and a multiple input shift register

Mirella Amelia V. Mioc[1]

Abstract – The aim of this paper is to propose some new techniques for the analysis of Linear Feedback Shift Registers LFSR and of Multiple Input Shift Registers MISR, based on their generating polynomials. First, mathematical representations of those polynomials ranks are found. Second, analytic expressions for their weights are discovered. Using these parameters the LFSR and MISR, associated to a given generating polynomial can be designed and simulated. Some simulation results proving the qualities of the new design, proposed in this paper, are presented.
Keywords: cryptography, feedback shift register, calculation.

I. INTRODUCTION

The Shift Register Cryptosystems' variant has been developed from the evolution of the encrypting techniques [2].
Such a cryptosystem is based upon generating a sequence in a finite field and for obtaining it a Feedback Shift Register is used.
The Linear Feedback Shift Registers are used in a variety of domains [3]: sequences generators; counters; BIST (Built-In-Self-Test) [4]; encryption; PRBS (Pseudo-Random Bit Sequences).
There are two types of LFSR from the utilization point of view:

- the well-known LFSR, that is a "in-tapping" LFSR;
- the "out-tapping" LFSR.

The "in-tapping" LFSR is usually called a MISR (Multiple Input Shift Register).
Cycle codes belong to algebraically codes for errors detecting.
All arithmetical operations will be developed in a Galois group.
This paper develops an analysis of a Linear Feedback Shift Register and a Multiple Input Shift Register.

II. THE DESCRIPTION OF THE EXPERIMENT

First of all, the algorithm was applied using grade 4 polynomials.
The results were correct and accurate.
Then the next natural step was to change the grade of the polynomial used for generating the scheme used in the calculus process. The chosen grade was 8, [1].
The Rijndael algorithm (chosen as replacement for the well known DES) is based on the implementing with shift registers of a method based on working in a Galois field, with the generator polynomial an irreducible grade 8 polynomial, thus substantially improving security [1].
Out of the programs, we came to the conclusion that the results should also rely on the previous link (given by the "equal with one" coefficient of the polynomial).

[1] Facultatea de Automatica si Calculatoare, Departamentul de Calculatoare, Bd. V, Parvan Nr. 2, 300223, Timisoara, e-mail: mmioc@cs.utt.ro

This "specific" link that is also part of the calculation for MISR is to be taken after the XOR was made for that rank of the polynomial.
The mathematically representation of each rank of the polynomial was made accordingly to these "specific" links.
Also, this new revealed thing was verified with the help of programs.
In the end, correlating the results obtained for the grade 4 polynomials with the results obtained for the 8 grade polynomials, we came to mathematical relations for calculating each rank.
The knowledge of the generating polynomial rank is very important in the estimation of the security level of a cryptosystem.
These relations point out the previous existing links and act similarly to a feedback "calculated" also from the previous links.
The link acts after the XOR was calculated and in this way takes the result that was previously obtained.
In order to demonstrate that those presented above are correct and precise, we made an analysis for all the 30 irreducible grade 8 polynomials that were also found in another program made by us, coming to three particular cases.
Out of this analysis, there was made a generalization, that led to the writing of specific programs for each irreducible grade 8 polynomial, used for the substantially improvement of security [1].
The simulation programs were tested in both ways: with the method of making tables according to the proposed circuits and also with mathematical methods that materialize the hard operations.
As input data sets were used several different multiple combinations of bits, randomly generated.
The two programs are described in this paper, one of them simulating the functioning of a LFSR and the functioning of an analytic MISR, and the other one simulating the functioning of a synthetic MISR. The programs are based on irreducible grade 8 polynomials, allowing the user to introduce the coefficients of the chosen polynomial.
There have been chosen three polynomials for testing these two programs.
The first one is the polynomial $P(x)=x^8+x^6+x^5+x^3+1$; the coefficients would be introduced in the program as it follows: 1 0 1 1 0 1 0 0 1, and would lead to the following scheme depicted in "Fig. 1.":

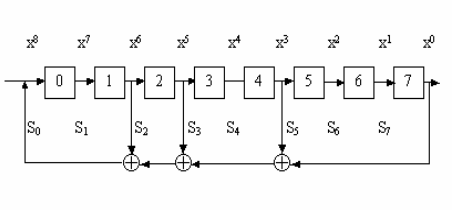

Fig. 1. Scheme for the polynomial $P(x)=x^8+x^6+x^5+x^3+1$

In the program the weights for each chosen polynomial are calculated.
The knowledge of those weighs is necessary for the design of the LFSR and MISR associated to the corresponding generating polynomial.
For the case in the "Fig. 1." Scheme, the weights are:
$S_0=1\ P(x)$
$S_1=x\ P(x)$
$S_2=x^2\ P(x)$
$S_3=(x^3+x)\ P(x)$
$S_4=(x^4+x^2+x)\ P(x)$
$S_5=(x^5+x^3+x^2)\ P(x)$
$S_6=(x^6+x^4+x^3+x)\ P(x)$
$S_7=(x^7+x^5+x^4+x^2)\ P(x)$
The second one is the polynomial $P(x)=x^8+x^4+x^3+x+1$; the coefficients would be introduced in the program as it follows: 1 0 0 0 1 1 0 1 1, and would lead to the following scheme depicted in "Fig. 2." :

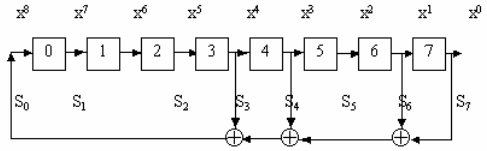

Fig. 2. Scheme for the polynomial $P(x)=x^8+x^4+x^3+x+1$

In the program the weights for each chosen polynomial are calculated.
For the case in the "Fig. 2." scheme, the weights are:
$S_0=1\ P(x)$
$S_1=x\ P(x)$
$S_2=x^2\ P(x)$
$S_3=x^3\ P(x)$
$S_4=x^4\ P(x)$
$S_5=(x^5+x)\ P(x)$
$S_6=(x^6+x^2+x)\ P(x)$
$S_7=(x^7+x^3+x^2)\ P(x)$

The third one is the polynomial $P(x)=x^8+x^7+x^6+x^4+1$; the coefficients would be

introduced in the program as it follows: 1 1 1 0 1 0 0 0 1, and would lead to the following scheme depicted in "Fig.3.":

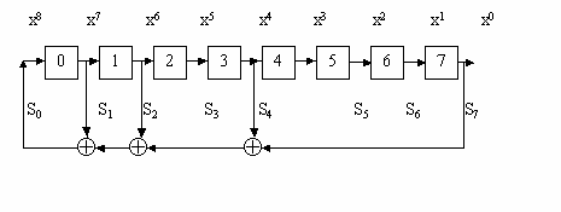

Fig. 3. Scheme for the polynomial $P(x)=x^8+x^7+x^6+x^4+1$

In the program the weights for each chosen polynomial are calculated.
For the case in "Fig. 3." scheme, the weights are:
$S_0=1$ P(x)
$S_1=x$ P(x)
$S_2=(x^2+x)$ P(x)
$S_3=(x^3+x^2+x)$ P(x)
$S_4=(x^4+x^3+x^2)$ P(x)
$S_5=(x^5+x^4+x^3+x)$ P(x)
$S_6=(x^6+x^5+x^4+x^2)$ P(x)
$S_7=(x^7+x^6+x^5+x^3)$ P(x)
In order to get the final results from the programs, the user has to introduce in the program the polynomial's coefficients as described above and also the input data sets, consisting of 8 columns, each of them having a length of 2^n elements (where n is an integer).
In the MISRS.CPP program the synthetic MISR is calculated for the input data given by the user and then the results are provided in the end.
Calculating the results consists of categorizing the 8 SRi steps in which the scheme will be treated in 3 major types of ways according to the below described types:

1. the first type is characterized by the existence of the coefficient of the i power corresponding to the SRi; in this case the procedure prel_SA is called, having i+1 as an argument;
2. the second type is characterized by the absence of the coefficient of the i power corresponding to the SRi; in this case the procedure prel_SN is called, having i+1 as an argument;
3. the third type is actually a particular one: it refers to the case of SR0, and the procedure prel_SZ is called, having no argument.

The prel_SZ procedure is reproduced below:

```
void prel_SZ()//SR0 calculation
{ int i,j,xor=0;
  for (j=1;j<=n;j++)
  { xor=0;
    for (i=0;i<8;i++)
    { if (ax[i]==1) //there is a //"connection"
                xor=xor^sr[i][j-1];
              if (i!=0)
                sr[i][j]=sr[i-1][j-1];
             }
    sr[0][j]=xor^col[0][j-1];
   }
  for (i=0;i<8;i++)
    rez[i][0]=sr[i][n];
 }
```

In this procedure, the result was calculated by translating the elements corresponding to the SRis, with i from 1 to 7, and by "collecting" the feedback in the SR0 while executing XOR with all the SRis that have a corresponding "connection" (the power i exists in the chosen polynomial).
The prel_SN procedure is reproduced below:

```
void prel_SN(int ind)
{ int i,j,xor=0;
  for (i=0;i<8;i++)
            for(j=0;j<51;j++)        sr[i][j]=0;
  for (j=1;j<=n;j++)
  { xor=0;
    for (i=0;i<8;i++)
    { if (ax[i]==1)
                xor=xor^sr[i][j-1];
              if (i!=0)
                sr[i][j]=sr[i-1][j-1];
             }
    sr[ind][j]=col[ind][j-1] ^sr[ind-1][j-1];
    sr[0][j]=xor;
   }
  for (i=0;i<8;i++)
    rez[i][ind]=sr[i][n];
 }
```

In this procedure, the result was calculated by translating the elements corresponding to the SRis, with i from 1 to 7 without ind (the argument of the procedure), by "connecting" the feedback in the SR0 while executing XOR with all the SRis that have a corresponding "connection" (the power i exists in the chosen polynomial) and by executing XOR with the element of the corresponding column and the element of the corresponding SRind.
The prel_SA procedure is reproduced below:

```
void prel_SA(int ind)
{ int i,j,k,xor=0;
  for (i=0;i<8;i++)
            for(j=0;j<51;j++) sr[i][j]=0;
```

```
 for (j=1;j<=n;j++)
 { xor=0;
  for (i=0;i<8;i++)
  { if ((ax[i]==1)&&
      ((ind-1)!=i))
           xor=xor^sr[i][j-1];
         if (i!=0)
           sr[i][j]=sr[i-1][j-1];
         }
  sr[ind][j]=col[ind][j-1]
^sr[ind-1][j-1];
  sr[0][j]=xor^sr[ind][j];
  }
 for (i=0;i<8;i++)
  rez[i][ind]=sr[i][n];
 }
```

In this procedure, the result was calculated by translating the elements corresponding to the SRis, with i from 1 to 7 without ind (the argument of the procedure), by "connecting" the feedback in the SR0 while executing XOR with all the SRis that have a corresponding "connection" (the power i exists in the chosen polynomial) and with the current corresponding SRind, and by executing XOR with the element of the corresponding column and the element of the corresponding SRind.

The difference between prel_SN and prel_SA is given by the SRind: it is taken into consideration in the procedure prel_SA in SR0, and does not appear in the procedure prel_SN in SR0.

The final result is calculated by making XORs with all the partial results obtained in the 8 steps on each and every column of all the eight columns.

III. THE RESULTS' INTERPRETATION

In the MISRALF.CPP program the analytic MISR and LFSR are calculated for the input data given by the user and the results are provided in the end.

For the given polynomial, the program calculates in the procedure genr the corresponding 8 weights. These weights are a base for calculation MISR and LFSR also.

The whole procedure for the analytic MISR consists of 8 steps: for each step is considered the corresponding column which is multiplied by the corresponding weight, then it is divided by the chosen polynomial, again it is multiplied by x^8 and, finally, divided by the chosen polynomial.

The result obtained in this way is the result of the current step of the calculation for the analytical MISR.

For each of the eight cases, the result is obtained as described above.

The final result for the analytical MISR is obtained by making XORs with all the results of the 8 steps on each and every column of all the eight columns.

The interpretation of this final result is that the ones and zeroes (eight by the number) obtained are the coefficients of a polynomial. The grade of this polynomial may be any of those between seven and zero.

The whole procedure for the LFSR is simpler: it is obtained by making XOR with all the weights multiplied by the corresponding column, and then, using the result obtained (whose grade gives the number of rounds that are to be done) as o column corresponding to the SR0 and no other columns, translating all the elements without 0, and "collecting" the feedback in the 0 element executing XOR with all the other elements that have a corresponding "connection" (the power with that rank exists in the chosen polynomial).

In this case, the final result consisting of those eight figures (ones and zeroes) represents also the coefficients of a polynomial, just as in the case of MISR.

The results obtained from the MISRS.CPP program and the other two results obtained from the MISRALF.CPP program are the same, especially since they have been obtained from the same input data used for calculation that are equivalent.

These kinds of calculation for verifying the correctness of the results have been made primarily on paper.

IV. CONCLUSIONS

Analyzing the experiments already presented it can be observed that the simulation speed for the MISR is higher versus the simulation speed for the LFSR.

The mathematical expressions of generating polynomial weights, reported in this paper, were very useful for the simulation of the corresponding LFSR and MISR.

V. REFERENCES

[1]B. Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code in C", John Wiley and Sons, New York, 1996.

[2] H. Niederreiter, "A Public–Key Cryptosystem Based on Shift Register Sequences", Proceedings of EUROCRYPT'85, Linz, Austria 1985.

[3] Horowitz and Hill, "The Art of Electronics", 2nd edition, 1989.

[4] A. Al–Yamani II, "Logic BIST: Theory, Problems, and Solutions", Stanford University, RATS/SUM02, 2002.

[5]http://www.mail-archive.com/cryptography-digests@senator-bedfellow.mit.edu/msg02659. html

[6]http://www-2.cs.cmu.edu/afs/cs/project/pscico -guyb/realworld/www/slidesS03/crypto2.4up.pdf