Study of Using Shift Registers in Cryptosystems for Grade 8 Irreducible Polynomials

MIRELLA AMELIA MIOC

1Department of Computers Science
“Politehnica” University of Timisoara
Bd. V. Parvan nr. 2 Timisoara 300223
ROMANIA


Abstract: - Different schemes based on shift registers may be used for coding and decoding. The goal of this analyze is to prove that the Linear Feed-back Shift Register and Multiple Input-Output Shift Register have the same function. The conclusion of this paper is that for grade 8 irreducible polynomials the weights are calculated with a formula discovered in this experiment.

Key-Words: - Cryptosystem, shift registers, calculate, irreducible polynomials, simulate, Rijndael.

1 Introduction

Beginning with 2000 Rijndael [1] [2] cryptosystem is officially the Advanced Encryption System (AES) [7], [8].

The old DES (Data Encryption Standard) [3], [4] was broken from Electronic Frontier Foundation in 3 days. The two authors Joan Daemen and Vincent Rijman from Holland chose to use a Galois Field GF (2^8) with the following generator polynomial [7], [8].

\[ P(x) = x^8 + x^4 + x^3 + x + 1 \] (1)

All arithmetical operations will be developed in a Galois group.

The Shift Register Cryptosystems’ variant has been developed from the evolution of the encrypting techniques [5]. 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 [5]: sequences generators; counters; BIST (Built-In-Self-Test) [6]; encryption; PRBS (Pseudo-Random Bit Sequences).

There are two types of LFSR from the utilization point of view: the well-known LFSR, that is an “in-tapping” LFSR and 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.

This paper develops an analysis of a Linear Feedback Shift Register and a Multiple Input – output Shift Register.

2 Description

First of all, the algorithm was applied using grade 4 polynomials.

The results were accurate and correct.

For each polynomial it was necessary to create three programs:

- one for simulating the use of LFSR;
- one for another simulation with MISR;
- another one for verifying the correctitude of the previous result.

Initially a program was specially developed to obtain all the irreducible grade 8 polynomial, thus substantially improving security [3].

Out of the programs, it comes to the conclusion that the results should also rely on the previous link (given by the “equal with one” coefficient of the polynomial).

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, it comes to mathematical relations for calculating each rank. 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, was made an analysis for all the 30 irreducible grade 8 polynomials that were also found in another program specially developed for this purpose, coming to three particular cases.

Out of this analyze, 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, 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 two polynomials for testing these two programs.

The first one is the polynomial

\[ P(x) = x^8 + x^6 + x^5 + x^3 + 1 \]  

where the coefficients would be introduced in the program as it follows:

1 0 1 1 0 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.

For the case in the Fig. 1 the weights are:

\[ S_0 = 1 \ P(x) \]
\[ S_1 = x \ P(x) \]
\[ S_2 = x^2 \ P(x) \]
\[ S_3 = (x^3 + x) \ P(x) \]
\[ S_4 = (x^4 + x^2 + x) \ P(x) \]
\[ S_5 = (x^5 + x^3 + x^2) \ P(x) \]

The second one is the polynomial (1), used in Rijndael Cryptosystem [10]:

\[ P(x) = x^8 + x^4 + x^3 + x + 1 \]

For the case in the Fig. 2 the weights are:

\[ S_0 = 1 \ P(x) \]
\[ S_1 = x \ P(x) \]
\[ S_2 = x^2 \ P(x) \]
\[ S_3 = x^3 \ P(x) \]
\[ S_4 = x^4 \ P(x) \]
\[ S_5 = (x^5 + x) \ P(x) \]
\[ S_6 = (x^6 + x^2 + x) \ P(x) \]
\[ S_7 = (x^7 + x^3 + x^2) \ P(x) \]

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). Because the new cryptographic Algorithms uses longer keys, now is important to improve the analysis of the functioning for shift registers of 16 and 32 [11].

3 The Procedure

In the MISRSIN.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:

- the first type is characterized by the existence of the coefficient of the i power corresponding to the SRi; in this case the procedure prelexi is called, having i+1 as an argument;
- the second type is characterized by the absence of the coefficient of the i power corresponding to the SRi; in this case the procedure prelabs is called, having i+1 as an argument;
- the third type is actually a particular one: it refers to the case of SR0, and the procedure prelzero is called, having no argument.
In the following rows the prelexi procedure is presented:

```c
void prelexi(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];
}
```

This time the basis is also the translation of 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” 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 code of the procedure prelabs is:

```c
void prelabs(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];
}
```

Here is the same translation of the elements from SR1 to SR7 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 difference between prelabs and prelexi is given by the SRind: it is taken into consideration in the procedure prelexi in SR0, and does not appear in the procedure prelabs 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.

The prelzero procedure is reproduced below:

```c
void prelzero}//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 is calculated by translating the elements corresponding to the SRis, with i from 1 to 7, without ind (the argument of the procedure), by “collecting” the feedback in the SR0 while executing XOR with all the SRis that have a corresponding “connection” and with the current corresponding SRind.

The analytic MISR and LFSR are calculated for the input data given by the user and the results are provided in the end of MISRCALC.

For the given polynomial, the program calculates in the procedure genr the corresponding 8 weights.

The procedure genr is:

```c
void genr()
{ //se calculeaza legaturile pentru polinomul introdus
  for (int i=0;i<9;i++)
    for (int j=0;j<8;j++)
      s[j][i]=0;
  //S0 e intotdeauna 1 pentru ca 8-r=8 (r e 0) si x^8/x^8 e 1
  s[0][0]=1;
  //r=8 nu are sens
  for ( i=1;i<8;i++)
    impart(i);
}
```

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 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 a 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 MISRCALC.CPP program are the same, since they are the result of 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.

### 4 Test Results

The next table contains all the 30 polynomials and the results of the tests. For these tests the input data were provided random from a Pseudorandom Generator [9].

Tab. 1 The complete situation about all the 30 grade 8 irreducible polynomials

<table>
<thead>
<tr>
<th>No.</th>
<th>Polynomial</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$x^8+x^2+x^4+x+1$</td>
<td>00110010</td>
</tr>
<tr>
<td>2</td>
<td>$x^8+x^3+x^4+x^2+1$</td>
<td>01010000</td>
</tr>
<tr>
<td>3</td>
<td>$x^8+x^5+x^3+x+1$</td>
<td>01010101</td>
</tr>
<tr>
<td>4</td>
<td>$x^8+x^3+x^5+x^3+1$</td>
<td>11010010</td>
</tr>
<tr>
<td>5</td>
<td>$x^8+x^3+x^4+x^2+1$</td>
<td>11001100</td>
</tr>
<tr>
<td>6</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>00101100</td>
</tr>
<tr>
<td>7</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>01011000</td>
</tr>
<tr>
<td>8</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>11100010</td>
</tr>
<tr>
<td>9</td>
<td>$x^8+x^3+x^5+x^3+1$</td>
<td>10001000</td>
</tr>
<tr>
<td>10</td>
<td>$x^8+x^3+x^5+x^3+1$</td>
<td>01100110</td>
</tr>
<tr>
<td>11</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>01010011</td>
</tr>
<tr>
<td>12</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>10100110</td>
</tr>
<tr>
<td>13</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>11000010</td>
</tr>
<tr>
<td>14</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>11010110</td>
</tr>
<tr>
<td>15</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>11000100</td>
</tr>
<tr>
<td>16</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>01001010</td>
</tr>
<tr>
<td>17</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>10101000</td>
</tr>
<tr>
<td>18</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>11100010</td>
</tr>
<tr>
<td>19</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>11100001</td>
</tr>
<tr>
<td>20</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>11101010</td>
</tr>
<tr>
<td>21</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>01010110</td>
</tr>
<tr>
<td>22</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>10110010</td>
</tr>
<tr>
<td>23</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>01100110</td>
</tr>
<tr>
<td>24</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>10110100</td>
</tr>
<tr>
<td>25</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>11010101</td>
</tr>
<tr>
<td>26</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>01101001</td>
</tr>
<tr>
<td>27</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>00101110</td>
</tr>
<tr>
<td>28</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>00010010</td>
</tr>
<tr>
<td>29</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>11000011</td>
</tr>
<tr>
<td>30</td>
<td>$x^8+x^3+x^5+x^4+1$</td>
<td>10001111</td>
</tr>
</tbody>
</table>

### 5 Conclusion

The analysis described in this paper proves that a MISR has always the same results as a LFSR.

A very useful observation was made in this process, that is calculating a MISR takes less time than calculating a LFSR.

The conclusion is that, instead of using a LFSR, there is the better and faster possibility of using a MISR.

In the case of using MISR, the speed increases very much, and it is well known that in using registers in cryptography an important goal is increasing the speed of calculations.

Also the aspect of security, as described in [3], was taken into consideration: the used polynomials are all irreducible grade 8 polynomials.

Another important aspect presented in this paper is the discovery of the new formula for the calculation of the weights used for obtaining the final result of MISR.

This formula was tested in the two above presented programs and the final conclusion is that the mathematical relations discovered are correct.
References: