A Synoptic of Software Implementation for Shift Registers Based on 16th Degree Primitive Polynomials

Mirella Amelia Mioc
Integrated Center for research, development and innovation in
Advanced Materials, Nanotechnologies, and Distributed Systems – MANSiD,
Stefan cel Mare University of Suceava, România
mmioc@cs.upt.ro

Abstract

Almost all of the major applications in the specific Fields of Communication used a well-known device called Linear Feedback Shift Register. Usually LFSR functions in a Galois Field GF(2^n), meaning that all the operations are done with arithmetic modulo n degree Irreducible and especially Primitive Polynomials. Storing data in Galois Fields allows effective and manageable manipulation, mainly in computer cryptographic applications. The analysis of functioning for Primitive Polynomials of 16th degree shows that almost all the obtained results are in the same time distribution.

Keywords*:* Cryptosystem, Irreducible polynomials, Pseudo-Random Sequence, Primitive Polynomials, Shift Registers.

1. Introduction

A code-breaking machine appeared as one of the first forms of shift register early in the 40's, in Colossus. It was a five-stage device built of vacuum tubes and thyratrons. Many different implementation forms were developed along the years. The LFSR (Linear Feedback Shift Register) is the basis of the stream ciphers and it is the most often used one in hardware designs. A string of memory cells that stored a string of bits and a clock pulse can advance the bits with one position in that string. For each clock pulse the new bit in the string is produced using the XOR of certain positions. The basis of every LFSR is developed with a polynomial, which can be irreducible or primitive (Angheloiu et al., 1986; Schneier, 1996). A primitive polynomial satisfies some additional mathematical conditions and determines for the LFSR to have its maximum possible period, meaning (2n-1), where n is the number of cells of the shift register or the length.

LFSR can be built based on XOR (exclusive OR) circuits or XNOR (exclusive denied OR). The difference of status is, of course, that the equivalent status will be 1, where it was 0. For an n bits LFSR, all the registers will be configured as shift registers, but only the last significant register will determine the feedback. An n bits register will always have n + 1 signals.

Every LFSR works by taking the XOR of the selected bits in its internal state and any LFSR containing all zero bits will never move to any other state, so one possible state must be excluded from any cycle. A LFSR is composed of memory cells connected together as a shift register with linear feedback. In digital circuits a shift register is formed by flip-flops and EXOR gates chained together with a synchronous clock. Shift registers are a form of sequential logic like counters. Always the shift registers produce a discrete delay of a digital signal or waveform. Considering that a shift register has n stages, the waveform is delayed by n discrete clock times. Usually the naming of the shift register follows a type of convention shown normally in digital logic, with the least significant bit on the left. According to the communication protocol, the signals will be addressed, not the registers. There are n+1 signals for each n-bit register. Always the next state of an LFSR is uniquely determined from the previous one by the feedback network. Any LFSR will generate a sequence of different states starting with the initial one, called seed.

A feedback shift register is composed of:
- a shift register
- a feedback function.

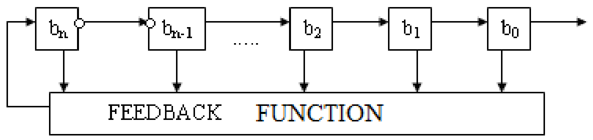

Figure 1. Basis scheme for a Feedback Shift Register

A LFSR can be represented as a polynomial of variable x referred to as the generator polynomial or the characteristic polynomial. The input bit is given from a linear function of the initial status for a special shift register called Linear Feedback Shift Register (LFSR). The initial value of the register is called seed and the produced sequence is completely determined by the initial status. Because the register has a finite number of possible statuses, after a period the sequence will be repeated. If the feedback function is very well chosen, the produced sequence will be random and the cycle will be very long, called by Golomb (1967) maximum lengths shift register sequences.

Goresky and Klapper (2004) show two possibilities to implement a LFSR:

- Fibonacci Form
- Galois.

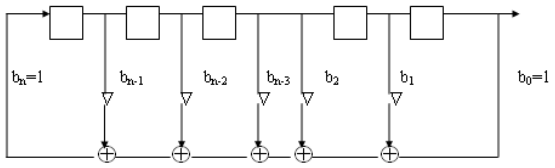

Figure 2. Fibonacci implementation

In Fibonacci form the weight for any status is 0, when there is not any connection and 1 for sending back. Exceptions make the first and the last one, both connected, so always on 1.

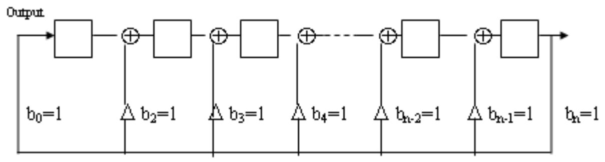

Figure 3. Galois implementation

In Galois implementation there is a Shift Register, whose content is modified each step at a binary value sent to the output. In Galois configuration the single shifted out bit is XOR ed with several bits in the shift register and in conventional configuration each new bit input to the shift register is the XOR of several bits in the register.

Comparing the two schemes of representation, it is shown that the weight order in Galois is opposite to the one in Fibonacci. From the hardware point of view, because of the reduced number of XOR gates in feedback, Galois implementation is faster than Fibonacci and so it is much more used. Some other names used for these two implementations are Simple Shift Register Generator (SSRG) for Fibonacci implementation and Multiple-Return Shift Register Generator (MRSRG) for Galois.

From the utilization point of view there are two kinds of LFSR: 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 experiment develops an analysis of a Linear Feedback Shift Register and a Multiple Input – output Shift Register.

By using a primitive polynomial in the polynomials modulo 2 as modular polynomial in the polynomial multiplication, can be created a Galois Field of order 2n with a polynomial beginning with xn. The most popular and widely used application of Galois Field is in Cryptography. Because all the data are represented as a vector in a finite field, encryption and decryption became easily to manipulate and straightforward by using mathematical arithmetic. Such kind of field can be denoted as GF(2n) or GF(n) and one of the famous applications for that is in the Rijndael Algorithm (AES), where n=8. Beginning with 2000, Rijndael cryptosystem is officially the Advanced Encryption System (AES) (Daemen & Rijmen, 2002). The old DES (Data Encryption Standard) was broken from Electronic Frontier Foundation in three days (Matsui, 1994). The two authors, Joan Daemen and Vincent Rijmen from Holland, chose to use a Galois Field GF (28) with the following generator polynomial.

$$P(x)=x^8+x^4+x^3+x+1$$

or '11B' in hexadecimal representation.

All arithmetical operations are developed in a Galois group. The Shift Register Cryptosystems variant has been developed from the evolution of the encrypting techniques (Schneier, 1996). Such a cryptosystem is based upon generating a sequence in a finite field, and for obtaining it a Feedback Shift Register is used.

There are some methods for using LFSR to build secure ciphers. For increasing the strength of the output from an LFSR it is often used another LFSR for controlling how often it is stepped. Another technique uses three LFSRs with different periods and it is known as the Geffe generator. It is usually necessary to combine the methods for obtaining more elaborate constructions. Almost all the shift registers applications representing generator polynomials need to be developed in a finite field.

Evariste Galois demonstrated that a field is an algebra with both addition and multiplication forming a group. Some ground information from Algebra demonstrated the importance of working with irreducible polynomials and primitive polynomials. Also the importance of using shift registers in cryptosystems based on irreducible polynomials is demonstrated in increasing the obtained security.

Applications for using The Linear Feedback Shift Registers are in a variety of Fields:

- Testing (Abramovici et al., 1990; Tsui, 1987);
- Pattern Generators;
- Optimized counters (Alfke, 1996);
- Checksums;
- Data Integrity;
- Data Encryption/ Decryption;
- Built-in Self-Test (BIST);
- Digital Signal Processing;
- Pseudo-random Number Generation (PN);
- Scrambler and/Descrambler;
- Signature Analyzer (Alvarez et al., 2008);
- Error Detection and Correction;
- Wireless communications.

2. Mathematical background

This finite field (FF) or Galois Field (GF) in abstract algebra is a field that contains only finitely many elements. Finite fields are important in algebraic theory, number theory, Galois theory, cryptography and coding theory (Berlekamp, 1968; Van Lint, 1992). It is possible to classify the finite fields by size. So, for each prime p and positive integer k there is exactly one finite field up to isomorphism of size pk. Each finite field of size q is the splitting field of the polynomial xq – x. A cyclic group is similarly the multiplicative group of the field. Finite fields have applications in many areas of mathematics and computer science, including coding theory and others.

The finite fields can be classified in the following rows:

a. The order or the number of elements of a finite field is of the form pn, where n is a positive integer and p is a prime number called the characteristics of the field,

b. For every prime number p and positive integer n there exists a finite field with pn elements,

c. Two finite fields with the same number of elements are isomorphic. It means that under the same remaining of the elements of one of these, both its addition and multiplication tables become identical to the corresponding table of the other one.

The use of a naming scheme for finite fields that specifies only the order of the field is justified by this classification.

Notations for a finite field can be: F_p^n and GF (p^n).

Arithmetic in a finite field is different from the standard integer arithmetic (Shannon, 1948). In the finite field there is a limited number of elements and the result of any operation performed is an element within that field. Each finite field is not infinite, but despite this there are infinitely many different finite fields, and their cardinal (number of elements) is necessarily of the form pn where p is a prime number and n is a positive integer. Two finite fields of the same size are isomorphic. The prime p is called the characteristic of the field and the positive integer n is called the dimension of this field over its prime field. Finite fields are used in a variety of applications as in the classical coding theory in linear block codes such as BCH (Bose Chaudhuri Hocquenghem) and RS (Reed Solomon) and in cryptography algorithms such as DES (Data Encryption Standard) and Rijndael encryption algorithm (AES).

A binary polynomial f(x) of degree n has the form:

$$f(x) = x^n + a_{n-1}x^{n-1} + \dots + a_1x + a_0$$

where a_i are binary coefficients.

Binary polynomials are added and multiplied in the normal manner of adding and multiplying polynomials, except that the resulting coefficients are reduced modulo two. A binary polynomial f(x) divides polynomial h(x) provided one can find a binary polynomial g(x) such that f(x)g(X)=h(x). A binary polynomial f(x) is Irreducible if its only divisors are 1 and f(x). An irreducible binomial polynomial on degree n is primitive if f(x) is not a divisor of xr+1 for any r less than

$$2^n-1.$$

The binary vector and power representations are two other methods of denoting GF(2^n). As before let f(x) be a primitive binomial polynomial of degree n. Considering z be a number such as f(z) = 0.

a. Binary Vector Representation

For each element $h(z) = a_0 + a_1z + ... + a_{n-1}z^{n-1}$ in GF($2^{n)}$ one can define a binary n-tuple by identifying:

$h(z)=\{a_0, a_1, ..., a_{n-1}\}$

b. Power Representation

It can be shown that since f(x) is a divisor of x2n-1 + 1 and not a divisor of xr + 1 for t less than 2n-1 then z2n-1 = 1 and that zi ≠ zj for i≤j≤2n-1. Using the exponential notation z0 = 1, GF(2n) can be defined in terms of zi as:

GF(2n) = { z0, z1, z3, .., z2n-2} U {0}

This is defined to be the power representation of GF(2n). Since every non-zero element in GF(2n) can be expressed as a power of z, this element is a Generator of GF(2n). For most applications of GF(2n) to cryptography, the value of n is large and it is impossible to construct a complete look-up table for the field. In transmission of data the binary n-tuple representation (a0, a1, ..., an-1) is used. The discrete log problem is that when the binary n-tuple representation of an element in GF(2n) is given and it will find its power representation. For security reasons it was demonstrated that the maximum number of pseudo-random sequences is obtained by using irreducible polynomials (Udar & Kagaris, 2007).

3. Experimental Results and Mathematical Calculus

The main subject of analysis for the functioning of linear feedback shift register (LFSR) and multiple input output shift register (MISR) has the irreducible or primitive polynomials for degree 4, 8 and 16 (Mioc, 2008). All the analysis is based on the three possible implementations for LFSR (Vlăduțiu & Crişan, 1989). First of all, there were developed programs for simulating the functioning of the three different types of implementations for comparing the obtained results for 4th degree irreducible polynomials (Mioc, 2009). Mioc (2008) shows a complete analysis and presentation of the functioning of LFSR for the 8th degree of irreducible polynomials. Basic information concerning the comparative analysis of a LFSR and MISR has been specified in Mioc (2005).

Table I. The 16th degree Primitive Polynomials

No	Octal	Binary
1	219913	10001000000001011
2	234313	10011100011001011
3	233303	10011011011000011
4	307107	11000111001000111
5	201735	10000001111011101
6	272201	10111010010000001
7	242413	10100010100001011
8	270155	10111000001101101
9	305667	11000101110110111
10	236107	10011110001000111
11	307527	11000111101010111
12	306357	11000110011101111
13	302157	11000010001101111
14	210205	10001000010000101

Table II. Some common 16th Degree Polynomials

(X^16+X^5+X^3+X^2+1) ;
(X^16+X^14+X^13+X^11+1 ;
(X^16+X^5+X^4+X^3+1) ;
(X^16+X^13+X^12+X^11+1) ;
(X^16+X^5+X^4+X^3+X^2+X+1) ;
(X^16+X^15+X^14+X^13+X^12+X^11+1) ;
(X^16+X^6+X^4+X+1) ;
(X^16+X^15+X^12+X^10+1) ;
(X^16+X^7+X^5+X^4+X^3+X^2+1) ;
(X^16+X^14+X^13+X^12+X^11+X^9+1) ;
(X^16+X^7+X^6+X^4+X^2+X+1) ;
(X^16+X^15+X^14+X^12+X^10+X^9+1) ;
(X^16+X^8+X^5+X^3+X^2+X+1 ;
(X^16+X^15+X^14+X^13+X^11+X^8+1) ;
(X^16+X^8+X^5+X^4+X^3+X^2+1) ;
(X^16+X^14+X^13+X^12+X^11+X^8+1;
(X^16+X^8+X^6+X^3+X^2+X+1) ;
(X^16+X^15+X^14+X^13+X^10+X^8+1) ;
(X^16+X^8+X^6+X^4+X^3+X^2+1) ;
(X^16+X^14+X^13+X^12+X^10+X^8+1) ;
(X^16+X^8+X^7+X^4+X^2+X+1;
(X^16+X^15+X^14+X^12+X^9+X^8+1;
(X^16+X^8+X^7+X^5+1) ;
(X^16+X^11+X^9+X^8+1) ;
(X^16+X^8+X^7+X^5+X^3+X^2+1) ;
(X^16+X^14+X^13+X^11+X^9+X^8+1) ;
(X^16+X^8+X^7+X^5+X^4+X^3+X^2+X+1) ;

A simulation program for the functioning on LFSR of the 16th degree for the Galois implementation was developed. In the following example an analysis for the 14 selected primitive polynomials will be presented. A list with the positions which will influence the future state is called tap sequence. For example, for the next scheme this is [16, 14, 13, 11].

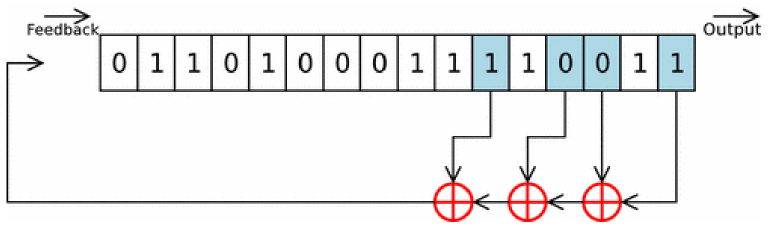

Figure 4. Scheme for the polynomial with [16, 14, 13, 11] tap sequence

This sequence can be represented by a polynomial mod 2, only with coefficients 1 and 0, called Feedback Polynomial or Characteristic Polynomial.

For the above scheme this polynomial is:

P(X)= X^16+X^14+X^13+X^11+1 ;

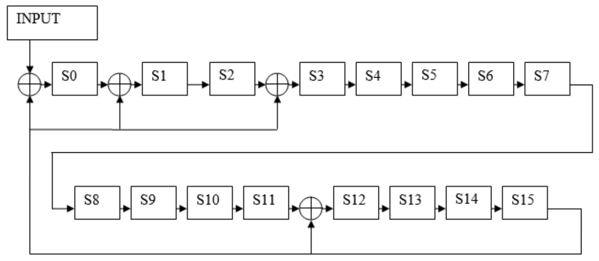

Figure 5. Galois Implementation for the Polynomial X^16+X^12+X^3+X+1

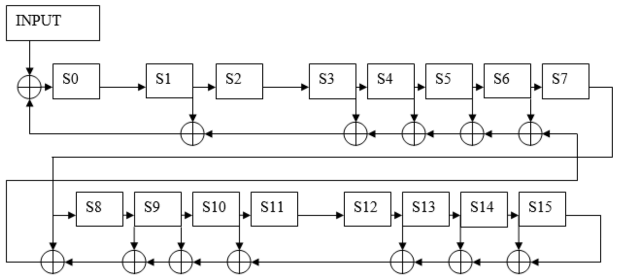

Figure 6. Fibonacci Implementation for the Polynomial X^16+X^12+X^3+X+1

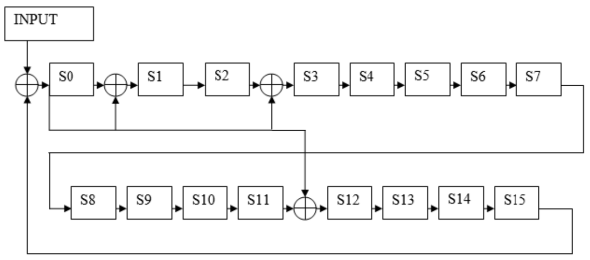

Figure 7. Ring Implementation for the Polynomial X6+X^12+X^3+X+1

Vlăduțiu and Crişan (1989) show three types of schemes for a 4th degree polynomial. Similar to it, there were developed the three different implementations for the Primitive Polynomial X^16+X^12+X^3+X+1. It can be specified that these schemes are according to the well-known Galois Form, Fibonacci representation and some other Forms that are rarely used, called Ring Implementation. All of these Implementations have the same function, because they describe a linear feedback shift register. In the experimental work there has been analyzed the behavior of 14 Primitive Polynomial degrees, 16 randomly selected. For each of these polynomials there has been developed the simulation of the specific functioning with a program. Because the goal of this experimental work was to compare the different obtained results, a few rows of input data of a different length have been selected.

First of all, in this analysis were verified the obtained times with the simulation program and they have been compared to each program. All the analyses submit the obtained results in the case of using Galois implementation. Times have been measured from 10 runs for the same string of input data and the average time has been calculated. For each Primitive Polynomial 6 different input data were used. In the following tables all the obtained results are shown by carrying out their time. So, for the 14 selected Polynomials and for the input data lengths of 20, 30, 40, 50, 100 and 1000 the corresponding times are presented (display). The following tables contain the time measured in seconds.

Table III. Results of the main program

	Time 20	Time 30	Time 40
Prel 1	0.00003471	0.00005425	0.00006220
Prel 2	0.00003864	0.00005288	0.00007746
Prel 3	0.00003761	0.00005422	0.00007319
Prel 4	0.00003562	0.00005112	0.00006738
Prel 5	0.00003662	0.00005137	0.00007063
Prel 6	0.00003763	0.00005172	0.00007026
Prel 7	0.00004377	0.00005093	0.00007052
Prel 8	0.00003557	0.00005649	0.00006912
Prel 9	0.000039022	0.000053175	0.00007273
Prel 10	0.00003513	0.00005102	0.00007181
Prel 11	0.00003442	0.00005404	0.00006725
Prel 12	0.00004232	0.00005098	0.00006978
Prel 13	0.00003514	0.00005345	0.00006794
Prel 14	0.00003489	0.00006187	0.00007052

Table IV. Results of the main program

	Time 50	Time 100	Time 1000
Prel 1	0.00023055	0.00022300	0.00173524
Prel 2	0.00010348	0.00022504	0.00215812
Prel 3	0.00008803	0.00022427	0.00224510
Prel 4	0.00008655	0.00022607	0.00242678
Prel 5	0.00009081	0.00023949	0.00211611
Prel 6	0.00008708	0.00022512	0.00226576
Prel 7	0.00008897	0.00021033	0.00216120
Prel 8	0.00008610	0.00021813	0.00209186
Prel 9	0.0001895	0.00209705	0.00348371
Prel 10	0.00008959	0.00028245	0.00209705
Prel 11	0.00008359	0.00028720	0.00214601
Prel 12	0.00008981	0.00023582	0.00203451
Prel 13	0.00008942	0.00023207	0.00214814
Prel 14	0.00008507	0.00021405	0.00236788

Table V. Coefficients number of the Generator Polynomials

Generator Polynomial	Coef no.
X^16+X^12+X^3+X+1	5
X^16+X^13+X^12+X^11+X^7+X^6+X^3+X+1	9
X^16+X^13+X^12+X^10+X^9+X^7+X^6+X+1	9
X^16+X^15+X^11+X^10+X^9+X^6+X^2+X+1	9
X^16+X^9+X^8+X^7+X^6+X^4+X^3+X^2+1	9
X^16+X^14+X^13+X^12+X^10+X^7+1	7
X^16+X^14+X^10+X^8+X^3+X+1	7
X^16+X^14+X^13+X^11+X^6+X^5+X^3+X^2+1	9
X^16+X^15+X^11+X^9+X^8+X^7+X^5+X^4+ X^2+X+1	11
X^16+X^13+X^12+X^11+X^10+X^6+X^2+X+1	9
X^16+X^15+X^11+X^10+X^9+X^8+X^6+X^4+ X^2+X+1	11
X^16+X^15+X^11+X^10+X^7+X^6+X^5+X^3+ X^2+X+1	11
X^16+X^15+X^10+X^6+X^5+X^3+X^2+X+1	9
X^16+X^12+X^7+X^2+1	5

In the following example, all the steps that simulate operation using the first Primitive Polynomial and the selected input data are showed. The used operations are shifting and XOR.

```
11010101010101010101
Checking input data: 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1

Initial Status 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
Step 0 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
Step 1 1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
Step 2 0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0
Step 3 1  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0
Step 4 0  1  0  1  1  0  0  0  0  0  0  0  0  0  0  0
Step 5 1  0  1  0  1  1  0  0  0  0  0  0  0  0  0  0
Step 6 0  1  0  1  0  1  1  0  0  0  0  0  0  0  0  0
Step 7 1  0  1  0  1  0  1  1  0  0  0  0  0  0  0  0
Step 8 0  1  0  1  0  1  0  1  1  0  0  0  0  0  0  0
Step 9 1  0  1  0  1  0  1  0  1  1  0  0  0  0  0  0
Step 10 0  1  0  1  0  1  0  1  0  1  1  0  0  0  0  0
Step 11 1  0  1  0  1  0  1  0  1  0  1  1  0  0  0  0
Step 12 0  1  0  1  0  1  0  1  0  1  0  1  1  0  0  0
Step 13 1  0  1  0  1  0  1  0  1  0  1  0  1  1  0  0
Step 14 0  1  0  1  0  1  0  1  0  1  0  1  0  1  1  0
Step 15 1  0  1  0  1  0  1  0  1  0  1  0  1  0  1  1
Step 16 1  0  0  0  0  1  0  1  0  1  0  1  1  1  0  1
Step 17 0  0  0  1  0  0  1  0  1  0  1  0  0  1  1  0
Step 18 0  0  0  0  1  0  0  1  0  1  0  1  0  0  1  1
Step 19 0  1  0  1  0  1  0  0  1  0  1  0  0  0  0  1
Runing Time: 0.00003421 seconds
```

The next two graphics show the obtained results from the execution of the main program for each of the 14 degrees, 16th primitive polynomials for three different situations depending on the lengths of the entrance data polynomial.

The lengths of the input polynomials were 20. 30. 40, 50, 100 and 1000 bits. The maximum number of sequences is 216-1(Solomon, 1967).

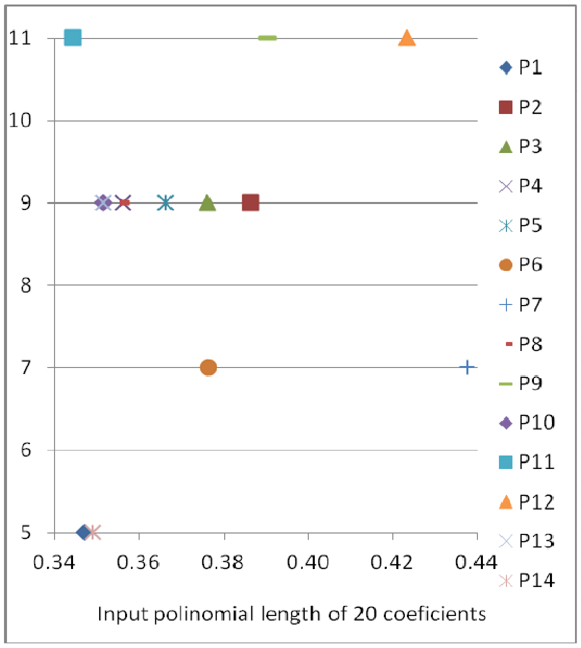

Figure 8. Graphic containing the results for 20 bits

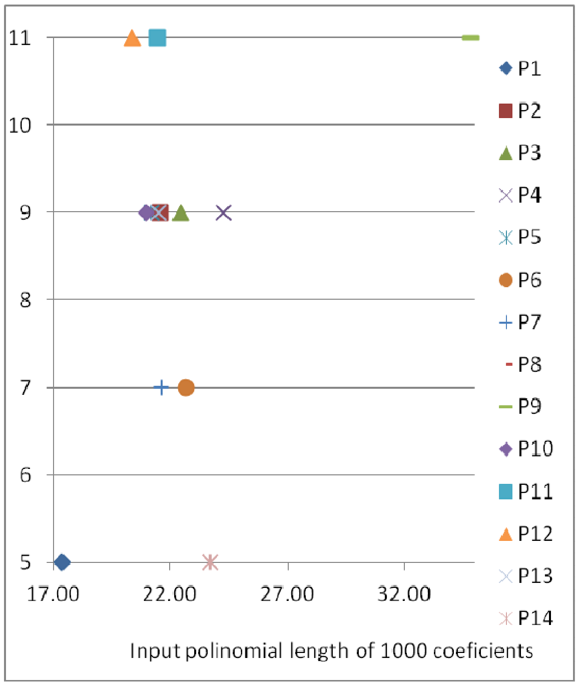

Figure 9. Graphic containing the results for 1000 bits

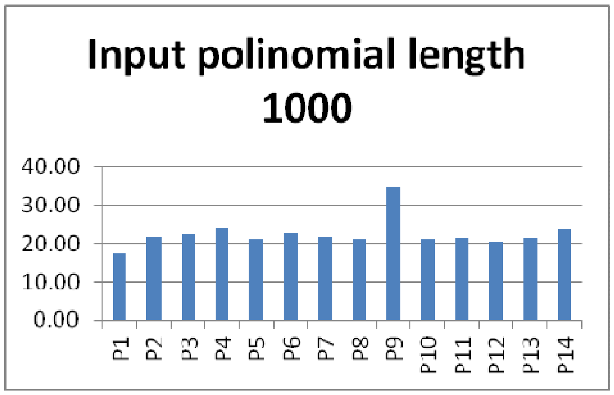

Figure 10. Graphic containing the results for 1000 bits

The distribution obtained depending on the length of the input string shows that time depends on the input length, but for lengths even closer together, the times are also close (this can be seen in Figure 8 for 20 bits inputs).

Time does not change so much depending on which of the 14 different 16th degree primitive polynomials has been used.

4. Conclusion

The entire analysis of functioning for primitive polynomials of 16th degree proves that almost all the obtained results are in the same distribution of time. The security aspect was taken into consideration, so that the used polynomials are all irreducible or primitive polynomials. A shift register is a device whose function is to shift its contents into adjacent positions within the register or, for the end position, out of the register. The main practical uses for a shift register are the conversion between parallel and serial data and the delay of a serial bit stream. The total number of the generated random state depends on the polynomial feedback.

LFSR based on the PN Sequence Generator is a technique used for different applications in Cryptography and also in the communication channel for designing encoder and decoder. This kind of analysis can be done from the hardware or software point of view. Panda et al. (2012) show an interesting simulation problem for long bit LFSR on FPGA referring 8, 16 and 32 Bits. Some similar problems are presented and analyzed in this paper. This study focuses on a comparative study of different types of implementations for a Linear Feed-back Shift Register for 16th degree primitive or irreducible polynomials. The results of all these experiments were used for obtaining some graphics showing the time distribution. From all the presented graphics it can be concluded that for 1000 bits length the best polynomial is the first one, meaning the polynomial number P1, and the worst is polynomial P9. All this analysis was preceded by another similar one on the same aspects linked with the 8th degree primitive polynomials.

5. Acknowledgement

This paper was developed and published as a part of the research for obtaining the PhD title in Stefan cel Mare University of Succeava, Romania. The infrastructure used for this work was partially supported by the project Integrated Center for research, development and innovation in Advanced Materials, Nanotechnologies, and Distributed Systems for fabrication and control, Contract No. 671/09.04.2015, Sectoral Operational Program for Increase of the Economic Competitiveness co-funded from the European Regional Development Fund.

References

Abramovici, M., Breuer, M. A., & Friedman, A. D. (1990). Digital Systems Testing and Testable Design, Computer Science Press;

Alfke, P. (1996). Efficient Shift Registers, LFSR, Counters, and Long Pseudo-Random Sequence Generators, XAPP 052, July 7,.

Alvarez, R., Martinez, F.-M., Vincent, J.-F., & Zamora, A. (2008). A Matricial Public Key Cryptosystem with Digital Signature, WSEAS TRANSACTIONS on MATHEMATICS Manuscript , vol.7,No.4 , pp. 195-204.

Angheloiu, I., Gyorfi, E., & Patriciu, V.V. (1986). Securitatea şi protecția informației în sistemele electronice de calcul, Ed. Militară, Bucureşti,.

Berlekamp, E. R. (1968). Algebraic Coding Theory, McGraw-Hill, New York.

Daemen, J. & Rijmen, V. (2002). "*The Design of Rijndael: AES - The Advanced Encryption Standard"*, Springer-Verlag.

Golomb, S.W. (1967). Shift Register Sequences, Holden-Day, San Francisco, Calif.

Goresky, M. & Klapper, A. (2004). Fibonacci And Galois Representations of Feedback with Carry Shift Registers, December 4;

Matsui, M. (1994). The First Experimental Cryptanalysis of the Data Encryption Standard. In Advances in Cryptology, *Proceedings of Crypto'94*, LNCS 839, Y. Desmedt, Ed., Springer-Verlag.

Mioc, M.A. (2009). Simulation study of the functioning of LFSR for grade 4 Irreducible Polynomials, WSEAS Conference ISPRA, 21-23 February.

Mioc, M.A. (2008). Study of Using Shift Registers in Cryptosystems for Grade 8 Irreducible Polynomials, *WSEAS Conference SMO*, 23-25 September.

Mioc, M.A. (2005). An analysis of functioning for a linear feed-back shift register and a multiple input-output shift register, Buletinul Stiintific al Universitatii „Politehnica" din Timisoara, Seria ELECTRONICA si TELECOMUNICATII, Transactions on electronics and communications, Tom 50(64), Fascicola 2.

Mioc, M.A. (2008). A complete analyze of using Shift Registers in Cryptosystems for Grade 4, 8 and 16 Irreducible Polynomials", WSEAS Transactions on Computers, Volume 7, Issue 10, ISSN: 1109-2750, October.

Panda, A.K., Rajput, P., & Shukla, B. (2012). FPGA Implementation of 8, 16 and 32 Bit LFSR with Maximum Length Feedback Polynomial Using VHDL, Rajkot, India.

Schneier, B. (1996). *Applied Cryptology: Protocols, Algorithms, and Source Code in C*, John Wiley and Sons, New York;

Shannon, C.E. (1948). A Mathematical Theory of Communication. Retrieved from http://worrydream.com/refs/Shannon%20-%20A%20Mathematical%20Theory%20of%20Communication.pdf

Solomon, G. (1967). Shift register sequences, Aegean Park Press, Laguna Hills, Canada.

Tsui, F. (1987) LSI/VLSI Testability Design, McGraw-Hill Book Company.

Udar, S. & Kagaris, D. (2007). LFSR Reseeding with Irreducible Polynomials, IOLTS, pp. 293-298

Van Lint, J.H. (1992). Introduction to Coding Theory, 2nd ed., Springer-Verlag, USA.

Vlăduțiu, M. & Crişan, M. (1989). Tehnica testării echipamentelor automate de prelucrare a datelor, Editura Facla, Timişoara.