Study of Software implementation for Linear Feedback Shift Register based on 8th degree irreducible polynomials

Mirella A. Mioc and Mircea Stratulat

Abstract—The Linear Feedback Shift Register is the simplest kind of feedback shift register. Based on the simple feedback sequences a large body of mathematical theory can be applied to analyzing LFSRs. A LFSR generates a random sequence of bits because it depends on the output feedback to the XOR gate. This property leads to generate pseudo-noise and pseudo-random number sequences and so LFSR are used in cryptography in data encryption and data compression circuits and also in communication and in error correction circuits. During the time a main problem was the speed. So, many research were develop in the frame of choosing the proper polynomial. This paper present an analysis for the 8^{th} degree Irreducible Polynomials from the point of view of time. The conclusion of this experiment is that almost all obtained results are in the same time distribution.

Keywords: Cryptosystem, Irreducible polynomials, Pseudo-Random Sequence, Shift registers.

I. INTRODUCTION

A code-breaking machine appeared as one of the first forms of a 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 most often used in hardware designs.

A string of memory cells stored a string of bits and a clock pulse can advance the bits with one position in that string.

For each clock pulse it is produced the new bit in the string using the XOR of certain positions.

The basis of every LFSR is developed with a polynomial, which can be irreducible or primitive.[4]

A primitive polynomial satisfies some additional mathematical conditions and determines for the LFSR to have its maximum possible period, meaning (2^n-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, 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.

An 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.

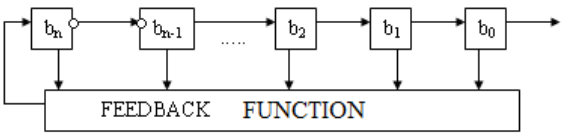

Fig.1. Feedback Shift Register Scheme

The most common type of shift registers used in cryptography are LFSR. A 4-bit LFSR tapped at the first and fourth bit is presented as functioning in the following .

The next sequence of internal states before repeating are produced when the initial value was 1111.

The total number of sequences are 15.

All of them are shown in the following rows:

1 1 1 1
0 1 1 1
1 0 1 1
0 1 0 1
1 0 1 0
1 1 0 1

0 1 1 0
0 0 1 1
1 0 0 1
0 1 0 0
0 0 1 0
0 0 0 1
1 0 0 0
1 1 0 0
1 1 1 0

The output sequence is the string of least significant bits:

1 1 1 1 0 1 0 1 1 0 0 1 0 0 0....

An LFSR can be represented as a polynomial of variable x referred to as the characteristic polynomial or the generator polynomial.

A LFSR is a shift register, whose input bit is given from a linear function of the initial status.

The initial value of the register is called seed and the sequence produced 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 good chosen the produced sequence will be random and the cycle will be very long.

There are two kinds of implementation for LFSR [7]:

- Fibonacci implementation
- Galois implementation.

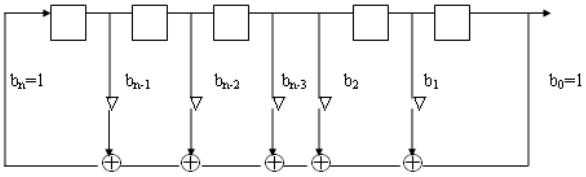

Fig. 2. Fibonacci implementation

In Fibonacci form the weight for any status is 0, when there isn't any connection and 1 for sending back.

Exceptions of this are the first and the last one, both connected, so always on 1.

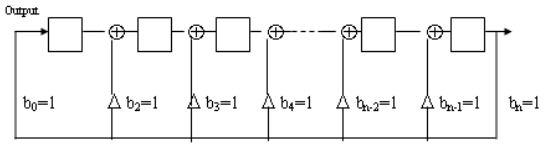

Fig.3. Galois implementation

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 bit shifted out is XORed 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 scheme of representation it is shown that the weight order in Galois is opposite the one in Fibonacci.

From the hardware point of view, Galois implementation is fastest than Fibonacci because of the reduced number of XOR gates in feedback and so it is much more used.

There are some industries in which Fibonacci form is referenced as SSRG (Simple Shift Register Generator) and Galois as MRSRG (Multiple-Return Shift Register Generator).

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. Another kind of classification for LFSR is internal and external. For each implementation there is the same characteristic polynomial defined by XOR positions. An important aspect is that this internal and external LFSRs with the same primitive polynomial do not generate same sequence, only same length.

An important point of view for an optimal use of LFSR is to choose primitive polynomials with minimum of XORs, because each gate produces its own delay and for increasing the speed is necessary to decrease their number. Some of the well known primitive polynomials with minimum of XORs are shown in the table 1.

Degree n	Polynomial Power for x
2,3,4,6,7,15,22	n,1,0
5,11,21,29	n,2,0
8,19	n,6,5,1,0
9	n,4,0
10,17,20,25,28	n,3,0
12	n,7,4,3,0
13,24	n,4,3,1,0
14	n,12,11,1,0
16	n,5,3,2,0

Table I. Primitive polynomials with minimum of XORs

LFSR is used for designing encoder and decoder for various communication channels and also for different cryptographic

applications. Some simulation programs were developed and used for testing and verifying the functioning of 4, 8 and 16 bit LFSR [11].

It is known that the total number of random state generated on LFSR depends on the characteristic or feedback polynomial.

By using maximum feedback polynomial it can obtain maximum 2^n-1. For 32 Bit LFSR the period is 4294967295, enough for most of the applications. The feedback polynomial can be expressed in finite field arithmetic as a polynomial mod 2,meaning that the coefficients must be 0 or 1.

Always the first and the last bits are connected as an input and output tap. When the number of LFSR taps is even, only then, LFSR will be maximum length.

From the synthesis results obtained for all 8-th, 16-th and 32-th degree polynomials the following can be concluded :

The total number of generated Random States for 8 bits are 255 and 65535 for 16 bits.

The 32 bits LFSR can generate in total 4294967295 Random States, but it takes a lot of time.

The large Random Sequence generated of 32 bits LFSR is more secure than the others obtained, but produce a lot of difficulties in practice, because the necessity of too much time.

So, in almost all cryptographic applications is sufficient to use 8 bit and 16 bit LFSR.

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 it can be created a Galois Field of order 2^n with a polynomial beginning with x^n.

Many encryption system are based on a hardware platform such a cipher using a LFSR generator. LFSR circuits are very fast, because to obtain the next states of each bit cell registry it is not necessary to have any combination functions, only input of individual flip-flop in directly connected to the output of the previous one.

A LFSR is capable to generate a maximum number of pseudo-random sequence only having a characteristic polynomial being also primitive polynomial.

Taking P(x) a primitive polynomial , the reciprocal polynomial is also primitive polynomial.

For being a primitive polynomial the polynomial must not have a common divisor of its coefficients greater than one.

P(x) polynomial is used in Galois implementation and the reciprocal polynomial in Fibonacci implementation.

Application from Communication systems to cryptography uses LFSRs as generators of pseudorandom sequences. Often LFSR software implementation are defined over the binary field GF(2).

The same software implementation over the extended fields GF(2^n) and in this case it will obtain an increase of speed.[24]

Such kind of field can be denoted as GF(2^n) or GF(n) and one of the famous applications for that is in the Rijndael Algorithm (AES), where n=8.

Beginning with 2000 Rijndael [5] cryptosystem is officially the Advanced Encryption System (AES).

The old DES (Data Encryption Standard) [9] was broken from Electronic Frontier Foundation in three 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.

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

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 [11]. 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, often it is 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.

Usually it is necessary to combine the methods for obtaining more elaborate constructions.

Almost all applications of using shift registers 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 security obtained.

The Linear Feedback Shift Registers are used in a variety of domains:

- Pattern Generators;
- Testing [1], [18];
- Optimized counters [2]
- Data Encryption/ Decryption;
- Built-in Self-Test (BIST) [7], [9];
- Digital Signal Processing
- Pseudo-random Number Generation(PN)
- Scrambler/Descrambler
- Data Integrity
- Checksums;
- Signature Analyzer [3];
- Error Correction;
- Wireless communications.

II. MATHEMATICAL BACKGROUND

A finite field (FF) or Galois Field (GF), so named in honour of Evariste Galois, 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 [16].
It's 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 p^k.

Each finite field of size q is the splitting field of the polynomial $x^q - x$.

Similarly the multiplicative group of the field is a cyclic group.

Finite fields have applications in many areas of mathematics and computer science, including coding theory [20] and others.

The finite fields are classified as follows:

- The number of elements or order, of a finite field is of the form p^n, where p is a prime number called the characteristics of the field, and n is a positive integer.
- There exists a finite field with p^n elements for every prime number p and positive integer n.
- Any two finite fields with the same number of elements are isomorphic. It means that under 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.

In the finite field there are 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 p^n 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 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 Rinjdael encryption algorithm (AES).

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

$$f(x) = x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$$

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). For example let $f(x) = x^3 + x + 1$ and $h(x) = x^7 + 1$, then from Example 3 f(x) divides h(x) since f(x)g(x) = h(x) where $g(x) = x^4 + x^2 + x + 1$.

A binary polynomial f(x) is said to be *irreducible* if its only divisors are 1 and f(x). For example one can show that $x^3 + x + 1$ is an irreducible polynomial. It can be shown that if f(x) is an irreducible binomial polynomial of degree n then f(x) is a divisor of $x^{2n-1} + 1$.

An irreducible binomial polynomial on degree n is primitive if f(x) is not a divisor of x^r+1 for any *r* less than 2^n-1. For example x^3+x+1 is a primitive polynomial since $x^3+ x + 1$ does not divide $x^r + 1$ for r less than 7.

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. Let z be a number such that f(z) = 0.

- Binary Vector Representation

For each element $h(z) = a_0 + a_1 z + \ldots + a_{n-1}z^{n-1}$ in $GF(2^n$ one can define a binary n-tuple by identifying:

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

- Power Representation

It can be shown that since f(x) is a divisor of $x^{2n-1} + 1$ and not a divisor of $x^r + 1$ for t less than 2^n-1 then $z^{2n-1} = 1$ and that $z^i \neq z^j$ for $i \leq j \leq 2^n-1$. Using the exponential notation $z^0 = 1$, $GF(2^n)$ can be defined in terms of z^i as:

$GF(2^n) = \{ z^0, z^1, z^3, .., z^{2n-2}\} \cup \{0\}$

This is defined to be the power representation of $GF(2^n)$. Since every non-zero element in $GF(2^n)$ can be expressed as a power of z this element is a *generator* of $GF(2^n)$.

For most applications of $GF(2^n)$ 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 $(a_0, a_1, \ldots, a_{n-1})$ is used. The discrete log problem is that, given the binary n-tuple representation of an element in $GF(2^n)$, find its power representation. For large n this is an intractable problem. The reverse problem of given the power representations find the binary n-tuple representation can be easily solved by using the division algorithm as follows:

- Let $a = z^i$ be an element of $GF(2^n)$ defined by primitive polynomial f(x);
- By division algorithm $x^i = q(x)f(x) + r(x)$ where degree of r(x) < n or 0;
- By substitution $z^i = q(z)f(z) + r(z)$ which implies that $z^i = f(z)$.

For security reasons it was demonstrated that the maximum number of pseudo-random sequences is obtained by using irreducible polynomials [19].

III. Experimental results and mathematical calculus

The main subject of analysis the functioning of linear feedback shift register (LFSR) and multiple input output shift register has the irreducible polynomials for degree 4, 8 and 16 [10].

All the analysis is based on the three possible implementations for LFSR [21].

First of all were developed programs for simulating the functioning for the three different types of implementations for comparing the obtained results for 4 degree irreducible polynomials [12].

For all analysis functioning of LFSR for 8 degree irreducible polynomials a complete presentation was made in [13].

In the following table there are presented all the 30 irreducible polynomials of 8 degree.

No.	Polynomial
1	$x^8+x^4+x^3+x+1$
2	$x^8+x^4+x^3+x^2+1$
3	$x^8+x^5+x^3+x+1$
4	$x^8+x^5+x^3+x^2+1$
5	$x^8+x^5+x^4+x^3+1$
6	$x^8+x^5+x^4+x^3+x^2+x+1$
7	$x^8+x^6+x^3+x^2+1$
8	$x^8+x^6+x^4+x^3+x^2+x+1$
9	$x^8+x^6+x^5+x+1$
10	$x^8+x^6+x^5+x^2+1$
11	$x^8+x^6+x^5+x^3+1$
12	$x^8+x^6+x^5+x^4+1$
13	$x^8+x^6+x^5+x^4+x^2+x+1$
14	$x^8+x^6+x^5+x^4+x^3+x+1$
15	$x^8+x^7+x^2+x+1$
16	$x^8+x^7+x^3+x+1$
17	$x^8+x^7+x^3+x^2+1$
18	$x^8+x^7+x^4+x^3+x^2+x+1$
19	$x^8+x^7+x^5+x+1$
20	$x^8+x^7+x^5+x^3+1$
21	$x^8+x^7+x^5+x^4+1$
22	$x^8+x^7+x^5+x^4+x^3+x^2+1$
23	$x^8+x^7+x^6+x+1$
24	$x^8+x^7+x^6+x^3+x^2+x+1$
25	$x^8+x^7+x^6+x^4+x^2+x+1$
26	$x^8+x^7+x^6+x^4+x^3+x^2+1$
27	$x^8+x^7+x^6+x^5+x^2+x+1$
28	$x^8+x^7+x^6+x^5+x^4+x+1$
29	$x^8+x^7+x^6+x^5+x^4+x^2+1$
30	$x^8+x^7+x^6+x^5+x^4+x^3+1$

Table II. The 8 degree irreducible polynomials

It was developed a simulation program for the functioning on LFSR of 8 degree for the Galois implementation [13].

In the following it will be presented an analysis for the irreducible polynomial:

$$P(x) = x^8+ x^6+ x^5+ x^3+1$$

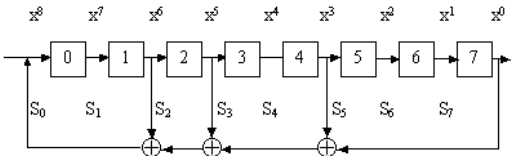

Fig.4. Fibonacci implementation for
$P(x) = x^8+ x^6+ x^5+ x^3+1$

The weights for each position are shown in the next rows:

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

First of all in this analysis it was verified with the simulation program for each weight the result, which is the same with the result of the polynomial division and also with the result obtained from the simulation table.

It is presented only the situation of the 7-th weight and after that the next table contains all the other results for all the weights.

All the analysis is referring the results obtained for the Galois implementation.

The result is obtained by the quotient division between the result of multiplying the remainder (the results from Fibonacci implementation) and x^8, and the used irreducible polynomial.

S_7

$(x^{15}+x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+x^9+x^8)(x^7+x^5+x^4+x^2)=$

$=x^{22}+x^{21}+x^{20}+x^{19}+x^{18}+x^{17}+x^{16}+x^{15}+x^{20}+x^{19}+x^{18}+x^{17}+x^{16}+x^{15}+x^{14}+x^{13}+x^{19}+$

$x^{17}+x^{16}+x^{15}+x^{14}+x^{13}+x^{12}+x^{17}+x^{16}+x^{15}+x^{14}+x^{13}+x^{12}+x^{11}+x^{10}=$

$=x^{22}+x^{21}+x^{19}+x^{18}+x^{14}+x^{13}+x^{11}+x^{10}$

$x^{22}+x^{21}+x^{19}+x^{18}+x^{14}+x^{13}+x^{11}+x^{10}$ | $x^8+x^6+x^5+x^3+1$

$x^{22}+x^{20}+x^{19}+x^{17}+x^{14}$ | $x^{14}+x^{13}+x^{12}+x^{11}+$

$x^{21}+x^{20}+x^{18}+x^{17}+x^{13}+x^{11}+x^{10}$ | $x^{10}+x^9+x^8+x^7+x^6+$

$x^{21}+x^{19}+x^{18}+x^{16}+x^{13}$ | x^5+x^3+x+1

$x^{20}+x^{19}+x^{17}+x^{16}+x^{11}+x^{10}$

$x^{20}+x^{18}+x^{17}+x^{15}+x^{12}$

$x^{19}+x^{18}+x^{16}+x^{15}+x^{12}+x^{11}+x^{10}$

$x^{19}+x^{17}+x^{16}+x^{14}+x^{11}$

$x^{18}+x^{17}+x^{15}+x^{14}+x^{12}+x^{10}$

$x^{18}+x^{16}+x^{15}+x^{13}+x^{10}$

$x^{17}+x^{16}+x^{14}+x^{13}+x^{12}$

$x^{17}+x^{15}+x^{14}+x^{12}+x^9$

$x^{16}+x^{15}+x^{13}+x^9$

$x^{16}+x^{14}+x^{13}+x^{11}+x^8$

$x^{15}+x^{14}+x^{11}+x^9+x^8$

$x^{15}+x^{13}+x^{12}+x^{10}+x^7$

$x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+x^9+x^8+x^7$

$x^{14}+x^{12}+x^{11}+x^9+x^6$

$x^{13}+x^{12}+x^8+x^7+x^6$

$x^{13}+x^{11}+x^{10}+x^8+x^5$

$x^{11}+x^7+x^6+x^5$

$x^{11}+x^9+x^8+x^6+x^3$

$x^9+x^8+x^7+x^5+x^3$

$x^9+x^7+x^6+x^4+x$

$x^8+x^6+x^5+x^4+x^2+x$

$x^8+x^6+x^5+x^2+1$

x^4+x+1

$(x^4+x+1)x^8=x^{12}+x^9+x^8$

$x^{12}+x^9+x^8$ | $x^8+x^6+x^5+x^3+1$

$x^{12}+x^{10}+x^9+x^7+x^4$ | x^4+x^2

$x^{10}+x^8+x^7+x^4$

$x^{10}+x^8+x^7+x^5+x^2$

$x^5+x^4+x^2$

	SR_0	SR_1	SR_2	SR_3	SR_4	SR_5	SR_6	SR_7
	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1
1	1	0	0	0	0	0	0	1
1	1	1	0	0	0	0	0	1
1	0	1	1	0	0	0	0	1
1	1	0	1	1	0	0	0	1
1	0	1	0	1	1	0	0	1
1	1	0	1	0	1	1	0	1
1	1	1	0	1	0	1	1	1
0	0	1	1	0	1	0	1	1
0	0	0	1	1	0	1	0	1
0	0	0	0	1	1	0	1	0
0	1	0	0	0	1	1	0	1
0	0	1	0	0	0	1	1	0
0	1	0	1	0	0	0	1	1
0	0	1	0	1	0	0	0	1
0	0	0	1	0	1	0	0	0
			x^2		x^4			

Table III. Calculus for S7

The weight	The result
7	$x^4+ x^2$
6	$x^6+ x^5+ x^4+ x^2+ 1$
5	$x^7+ x^6+ x^5+ x^4+ 1$
4	$x^7+ x^3+ x^2$
3	$x^7+ x^6+ x$
2	$x^7+ x^6+ x + 1$
1	$x^5+ x^4$
0	$x^6+ x^4+ x^2+ 1$

Table IV. The weights for Galois implementation for $P(x) = x^8+ x^6+ x^5+ x^3+1$

The next program was developed for analysis of all 8th degree irreducible polynomials. There are 30 different processings for each polynomial and in the main program it was counted the time for each situation. The program was executed in linux and it was necessary to use – lrt option for accessing time specific functions.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#include "prelucrareA0.c"
#include "prelucrareAl.c"
#include "prelucrareA2.c"
#include "prelucrareA3.c"
#include "prelucrareA4.c"
#include "prelucrareA5.c"
#include "prelucrareA6.c"
#include "prelucrareA7.c"
#include "prelucrareA8.c"
#include "prelucrareA9.c"
#include "prelucrareAlO.c"
#include "prelucrareAll.c"
#include "prelucrareAl2.c"
#include "prelucrareA13.c"
#include "prelucrareAl4.c"
#include "prelucrareA15.c"
#include "prelucrareA16.c"
#include "prelucrareAl7.c"
#include "prelucrareAl8.c"
#include "prelucrareA19.c"
#include "prelucrareA20.c"
#include "prelucrareA21.c"
#include "prelucrareA22.c"
#include "prelucrareA23.c"
#include "prelucrareA24.c"
#include "prelucrareA25.c"
#include "prelucrareA2 6.c"
#include "prelucrareA27.c"
#include "prelucrareA28.c"
#include "prelucrareA29.c"
struct timespec tstart={0,0}, tend={0,0};


FILE *pf,*fin;
int k,x[23],Lung;

void afis(int tab[],int n)
{
   int i;

   if(k==1)
   {
   printf("     ");
   fprintf(pf,"     ");
   }
   else
   {
      fprintf(pf,"x^%d.",Lung-k+1);
      printf("x^%d.",Lung-k+1);
   }

   for (i=0;i<n;i++)
   {
      printf(" %d ",tab[i]);
      fprintf(pf," %d ",tab[i]);
   }
   fprintf(pf,"\n");
   printf("\n");
}

typedef void (*prel_t)(int n, int knt, int a[],int s[]);

prel_t preltab[30] =
{prelucrareA0,prelucrareA1,prelucrareA2,prelucrareA3,prelucrareA4,prelucrareA5,prelucrareA6,prelucrareA7,prelucrareA8,
```

```
prelucrareA9,prelucrareA10,prelucrareA11,prelucrareA12,prel
ucrareA13,prelucrareA14,prelucrareA15,prelucrareA16,preluc
rareA17,prelucrareA18,prelucrareA19,prelucrareA20,
prelucrareA21,prelucrareA22,prelucrareA23,prelucrareA24,pr
elucrareA25,prelucrareA26,prelucrareA27,prelucrareA28,prel
ucrareA29};

int main(int argc, char *argv[])
{

   if(argc==1)
   {
   printf("Lipsa nume la functia prelucrare. (1 ,2 ,3 ...)\n");
   exit(EXIT_FAILURE);
   }
   else
   {

      int s[50],a[50],i,j,numprel;

      pf=fopen("LFSR8_Schema_A.txt","a");
      if(!pf)
      {
      printf("Eroare la deschiderea fisierului
LFSR8_Schema_A.txt !!!\n");
      exit(EXIT_FAILURE);
      }

      fin=fopen("lfsr8.txt","r");
      if(!fin)
      {
      printf("Eroare la deschiderea fisierului lfsr8.txt !!!\n");
      exit(EXIT_FAILURE);
      }

      for(i=0;i<8;i++)
      s[i]=0;

      if(fscanf(fin,"%d",&Lung)!=1)
      {
      printf("Eroare la citirea lungimii polinomului!\n");
      exit(EXIT_FAILURE);
      }

      for(i=0;i<Lung;i++)
         if(fscanf(fin,"%d",&x[i])!=1)
      {
         printf("Eroare la citirea indicelui %d\n",i);
         exit(EXIT_FAILURE);
      }

   numprel = atoi(argv[1]);

      clock_gettime(CLOCK_MONOTONIC, &tstart);
      k=1;

      printf("Se foloseste functia: %d!\n",numprel);
   fprintf(pf,"Se foloseste functia: %d!\n",numprel);

   for(i=0;i<Lung+1;i++)
      {
      afis(s,8);
      preltab[numprel-1](8, i, a, s);
         for(j=0;j<8;j++)
         s[j]=a[j];
      }

      clock_gettime(CLOCK_MONOTONIC, &tend);

      printf("Executarea buclei a durat: %.5f secunde.\n\n",
        ((double)tend.tv_sec + 1.0e-9*tend.tv_nsec) -
        ((double)tstart.tv_sec + 1.0e-9*tstart.tv_nsec));
      fprintf(pf,"Executarea buclei a durat: %.5f secunde.\n\n",
        ((double)tend.tv_sec + 1.0e-9*tend.tv_nsec) -
        ((double)tstart.tv_sec + 1.0e-9*tstart.tv_nsec));

      fclose(pf);
      fclose(fin);
      return 0;
   }
}
```

In this program it was used an input file (lfsr8.txt) containing the input data polynomial and another output file (LFSR8_Schema_A.txt) containing the output sequences while the time is measured for each execution of functioning simulation for all the 30 irreducible polynomials of 8th degree.

The following table contains the time measured in seconds.

	Time 10bits	Time 16 bits	Time 255 bits
prelucrareA0	0.00083	0.0002	0.02697
prelucareA1	0.00016	0.00022	0.00361
prelucareA2	0.00007	0.00016	0.00389
prelucareA3	0.00012	0.00021	0.00373
prelucareA4	0.00016	0.00015	0.00384
prelucareA5	0.00012	0.00022	0.02516
prelucareA6	0.00017	0.00013	0.00364
prelucareA7	0.00012	0.00024	0.00414
prelucareA8	0.0002	0.00022	0.0039
prelucareA9	0.0008	0.00019	0.00392
prelucareA10	0.00018	0.00021	0.00397
prelucareA11	0.00011	0.00016	0.00346
prelucareA12	0.00018	0.00016	0.00425
prelucareA13	0.00007	0.00019	0.00397
prelucareA14	0.00016	0.00022	0.00424
prelucareA15	0.00022	0.00021	0.00696
prelucareA16	0.00015	0.00023	0.00387
prelucareA17	0.00007	0.00022	0.00405

prelucareA18	0.00013	0.00016	0.00377
prelucareA19	0.00015	0.00015	0.00396
prelucareA20	0.00017	0.0002	0.00446
prelucareA21	0.00017	0.00021	0.01093
prelucareA22	0.00012	0.00016	0.00382
prelucareA23	0.00016	0.00015	0.00396
prelucareA24	0.00016	0.0002	0.00399
prelucareA25	0.00007	0.00017	0.00825
prelucareA26	0.00015	0.00025	0.00351
prelucareA27	0.00013	0.00022	0.00377
prelucareA28	0.00012	0.00021	0.00374
prelucareA29	0.0002	0.00019	0.00382

Table V. Results of the main program

The next two graphics show the obtained results from the execution of the main program for each of all 30 degree 8 irreducible polynomials for three different situations depending on the entrance data polynomial.

The lengths of the entrance polynomials were 10, 16, 255bits. The maximum length of sequences is 2^8-1 [17].

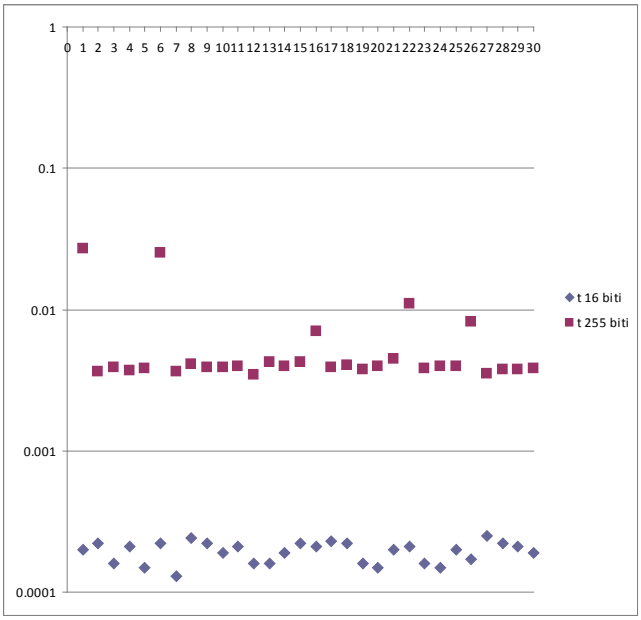

Fig.5. Time distribution for input data of 16, 255 bits

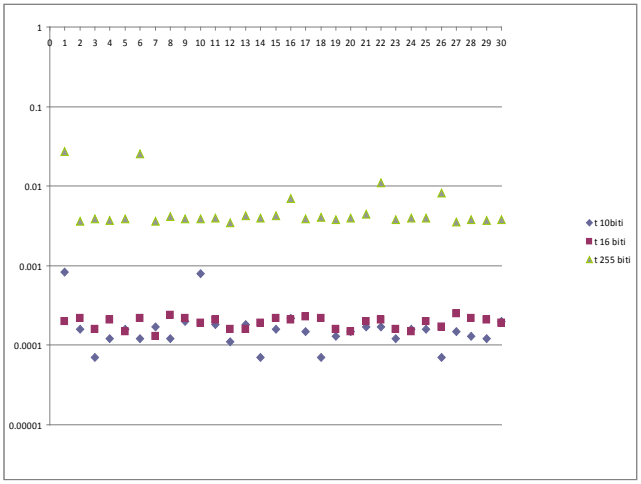

Fig.6. Time distribution for input data of 10, 16, 255 bits

The distribution obtaining in function of the lengths of data input polynomial shows that time depends of input length, but for lengths that are quit close the processing times obtain are also close (this can be seen in Fig.6 results for 10 and 16 bits inputs).

Time does not change so much depending on which one of the 30 different 8^{th} degree irreducible polynomials has been used.

IV. RELATED WORKS

A huge amount of digital data is being exchanged over unsecured channels because the rapid growth of computer networks and advances technology.

Information security has become a very critical point of modern computing systems to protect data from unauthorized access. Security in an important issue in communication and storage. Transmission and storage of multimedia data has increased in today's digital communication. The advanced Encryption Standard (AES), International Data Encryption Algorithm and some conventional algorithms like Data Encryption Standard have certain limitations in multimedia data encryption, so many new methods for encryption were developed by numerous researchers.

For protecting digital data from unauthorized eavesdropping three different ways can be used: cryptography, steganography and watermarking.

Cryptography has become one of the major tools for obtaining high level of security dealing with the development of techniques for converting information.

Yoon and Kim in 2010 developed a chaotic image cipher in witch initially a small matrix was generated using chaotic logistic map. Developing some conventional classical cryptographic techniques is the most common way to protect large multimedia fields.

Both software and hardware implementations of RSA or El-Gamal cannot support fast and high speed encryption rates. Triple DES or Blowfish are suitable for transmission of large amounts of information. Much more suitable for multimedia content rich data encryption are the chaos-based crypto schemes. V G. A. Sathishkumar, K. Bhoopathy proposed an Encryption Algorithm using Chaotic Block Cipher [23].

They proved that the proposed approach is very good to adeptly trade offs between the speed and protection. The dimension and complexity of real world data sets is in an continuous grown and determines combining two or more algorithms for obtaining advanced data mining algorithms.

A special place in this research is occupied of a powerful machine learning technique from the family of evolutionary algorithms called genetic programming [22].Due cryptography is the art of making ciphertext, cryptanalysis is the art of breaking them.

Vimalathithan R., M. L. Valarmathi developed an approach for breaking the key used in Simplified-Data Encryption Standard (S-DES) using Genetic Algorithm (GA) combined with two optimization techniques called Particle Swarm Optimization (PSO) and Genetic Swarm Optimization (GSO) [15].

Analyzing the different kinds of LFSR implementation a new problem appear. This problem is how LFSR parameters must be chosen for developing an efficient implementation [8].

C. Lauradoux obtained a new tool for generatingthek-bit leap-forward implementation of an LFSR with a given polynomial. Putting together the feedback computation and shift operation it results the leap-forwarding. This implementation uses a vector-matrix multiplication for showing a single step of an LFSR. The analyze uses both Fibonacci and Galois LFSR setups in VLSI design.

The conclusions proved that for increasing the security the characteristic polynomial must be primitive.

Design for Testability (DFT) is another area for using LFSR functions.

Built-In Self-Test (BIST) allows for a circuit itself testing without using any other external equipment.

So BIST are low cost comparing to any external testing.

Specially in BIST implementation LFSR can be used for pseudo-random patterns, response compaction, polynomial division and others. Also it is possible to use in the same BIST, a LFSR for implementing the pseudo-random test – pattern generator and a Multi-Input Shift Register (MISR) for a signature analyzer.

LFSR is very popular for both implementation of Test Pattern Generator (TPG) and Output Response Analyzer (ORA).

The popularity of LFSR used in BIST application is owed to compact and simple design.

V. CONCLUSION

The whole analysis of functioning for irreducible polynomials of 8 degree proves that almost all obtained results are in the same distribution of time.

The aspect of security was taken into consideration, so that the used polynomials are all irreducible or primitive polynomials.

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

This formula was tested for all the situations referring to degree 8 irreducible polynomials and the final conclusion is that the mathematical relations discovered are correct.

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 delay of a serial bit stream;
- the convert between parallel and serial data.

This study focuses on a comparative study of different types of implementations for a Linear Feed-back Shift Register for 8 degree irreducible polynomials. The results of all these experiments were used for obtaining some graphics showing the time distribution.

REFERENCES

[1] Abramovici M., Breuer M. A., Friedman A. D., Digital Systems Testing and Testable Design, Computer Science Press,1990;

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

[3] Alvarez R., Martinez F.-M., Vicent J.-F., Zamora A., A Matricial Public Key Cryptosystem with Digital Signature, *WSEAS TRANSACTIONS on MATHEMATICS* Manuscript , vol.7,No.4,2008, pp. 195-204.

[4] Angheloiu I., Gyorfi E., Patriciu V.V., Securitatea şi protecţia informaţiei în sistemele electronice de calcul, Ed. Militară, Bucureşti, 1986.

[5] Daemen J., Rijmen V., "*The Design of Rijndael: AES - The Advanced Encryption Standard*", Springer-Verlag, 2002.

[6] Delgado-Mohatar O., Fúster-Sabater A., Software Implementation of Linear Feedback Shift Registers over Extended Fields, International Joint Conference CISIS'12-ICEUTE´12-SOCO´12 Special Sessions, Advances in Intelligent Systems and Computing Volume 189, 2013, pp 117-126.

[7] Goresky M., Klapper A., Fibonacci And Galois Representations of Feedback with Carry Shift Registers, December 4, 2004;

[8] Lauradoux C. From Hardware to Software Synthesis of Linear Feedback Shift Registers, Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International, Long Beach, CA, 26-30 March 2007.

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

[10] Mioc M. A., An analyze 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, 2005.

[11] Mioc M. A., A complete analyze of using Shift Registers in Cryptosystems for Grade 4, 8 and 16 Irreducuble Polynomials, WSEAS Transactions on Computers, Volume 7, October 2008, pp 1805-1817

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

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

[14] Owais S., Krömer P., Platoš J., Snášel V., Zelinka I., Data Mining by Symbolic Fuzzy Classifiers and Genetic Programming–State of the Art and Prospective Approaches, WSEAS TRANSACTIONS on COMPUTERS, Issue 6, Volume 12, March 201385-94.

[15] Sathishkumar G. A., Bhoopathy K., A Novel Image Encryption Algorithm Using Pixel Shuffling and BASE 64 Encoding Based Chaotic Block Cipher (IMPSBEC) WSEAS TRANSACTIONS on COMPUTERS, Issue 6, Volume 10, June 2011,pp 169-178.

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

[17] Shannon C.E., Mathematical Theory of Communication, 1948.

[18] Solomon G., Shift register sequences, Aegean Park Press, Laguna Hills, Canada, 1967.

[19] Tsui F., *LSI/VLSI Testability Design*, McGraw-Hill Book Company, 1987.

[20] Udar S., Kagaris D., LFSR Reseeding with Irreducible Polynomials, *IOLTS 2007*, pp. 293-298.

[21] Van Lint J.H., Introduction to Coding Theory, 2nd ed., Springer-Verlag, USA, 1992.

[22] Vlăduţiu M., Crişan M., Tehnica testării echipamentelor automate de prelucrare a datelor, Editura Facla, Timişoara, 1989.

[23] Vimalathithan R., M. L. Valarmathi, Cryptanalysis of Simplified-DES using Computational Intelligence,), WSEAS TRANSACTIONS on COMPUTERS, Issue 6, Volume 10, July 2011,pp 210-219.

[24] Yoon J. W, Hyoungshick K., An image encryption scheme with a pseudorandom permutation based on chaotic maps, Communications in Nonlinear Science and Numerical Simulation Volume 15, Issue 12, December 2010, Pages 3998– 4006.