Global Journal of Computer and Engineering Technology (GJCET)

Vol. 1, No. 1, January, 2021 ISSN: 2767-1933

MATRIX APPROACH FOR MAXIMUM PERIOD LINEAR FEEDBACK SHIFT REGISTER STREAM CIPHERS*

Daniel Okunbor, Fayetteville State University, U.S.A. (diokunbor@uncfsu.edu)

ABSTRACT

Maximum-period LFSRs are of interest in cryptography since they generate pseudo-random bit sequences that are large enough for them to be computationally secure using exhaustive search methods. A maximum-period LFSR has better linear complexity and satisfies the Golomb randomness postulates, and the autocorrelation function has two values only. The number of maximum-period LFSRs of degree *n* grows super exponentially requiring multiprocessor computing systems. This paper attempts to develop matrix computing constructs for generating maximum-length LFSRs using commodity systems.

Keywords: Cryptography, Ciphers, Pseudo-Random, Linear Feedback Shift Registers

*The work is supported in part by a grant (Award No: CNS200239) from the Division of Computer and Network Systems of the National Science Foundation through the Historically Black Colleges and Universities Undergraduate Programs Excellence in Research.

1. INTRODUCTION

A cryptographic algorithm is used to encrypt and decrypt data of several types (texts, images, audio, and videos) and we shall refer to the combined encryption and decryption algorithms as a cipher. Ciphers can be classified into two categories, namely, symmetric (also called private key) and asymmetric (also called public key) ciphers. A symmetric cipher requires a single secret key for both encryption and decryption, while asymmetric cipher requires two secret keys, one is a public key used for encryption and the other is a private key for decryption.

Symmetric ciphers are further classified as stream and block ciphers. A stream cipher encrypts and transmits data bits or characters one bit or character at a time, while a bock cipher encrypts and transmits data bits or characters in blocks or groups of bits. Since each bit in stream ciphers is independently encrypted and decrypted, they do not suffer from propagation errors as in block ciphers and they tend to have better software efficiency (Galanis, et.al., 2005).

The use of stream ciphers dates to the early 1900s when in 1917, Gilbert Vernam invented the electromechanical machine that automatically encrypted teletypewriter communication (Borowski & Leśniewicz, 2012). A stream cipher is based on a key stream of bits($s_i \in \{0,1\}, \forall i$). The key stream is XORed with plaintext to obtain the ciphertext $y_i = s_i \oplus x_i$ and for decryption, it is XORed with ciphertext to obtain the plaintext $x_i = s_i \oplus y_i$, where $x_i, y_i \in \{0.1\}$ represent the bits of the plaintext and ciphertext, respectively . Stream ciphers have applications in communications over the Internet, mobile networks, image transmissions and other embedded systems.

This paper addresses the linear feedback shift registers that are used as stream ciphers for generating pseudo-random sequences. In particular, we will utilize matrix operations to determine maximum period linear feedback shift registers which are considered to give computationally secured pseudo-random sequences.

2. LINEAR FEEDBACK SHIFT REGISTERS (GALOIS)

A linear feedback shift register (LFSR) is one that is characterized by a linear recurrence consisting of connection vector and initial state vector denoted by

$$\boldsymbol{c} = (c_0, c_1, \dots, c_{n-1}) \text{ and } \boldsymbol{s} = (s_0, s_1, \dots, s_{n-1}),$$

respectively. Here, $c_i, s_i \in \{0,1\}$.

We say that the LFSR is of degree n. The output sequence of the LFSR is denoted by

$$\{s_0, s_1, s_2, \dots, s_{n-1}, s_n, s_{n+1}, \dots\}, s_i \in \{0,1\}.$$

At any given time, the internal state values must be shifted to the left (or right, depending on your choice of direction of the shift). We assume right shift in our implementation. Denoting the initial state values (t=0) by

$$\boldsymbol{s}(0) = \big(s_0(0), s_1(0), s_2(0), \dots, s_{n-1}(0)\big),$$

the state value at *t* epoch is given by

$$\boldsymbol{s}(t) = \big(s_0(t), s_1(t), s_2(t), \dots, s_{n-1}(t)\big).$$

Using the popular Galois representation of LSFR, named after the French Mathematician Evariste Galois, see (Weisstein, undated; Paarand Pelzl, 2010; Sachs, 2017), the state is updated using the following sequence:

$$\begin{aligned}
&s_0(t) = c_0 s_{n-1}(\mathrm{t}-1)\\
&s_1(t) = c_1 s_0(t-1) \oplus s_{n-1}(t-1)\\
&\dots\\
&s_{n-2}(t) = c_{n-2} s_{n-3}(t-1) \oplus s_{n-1}(t-1)\\
&s_{n-1}(t) = c_{n-1} s_{n-2}(t-1) \oplus s_{n-1}(t-1)
\end{aligned}$$

The matrix representation is

$$\begin{bmatrix} s_0(t) \\ s_1(t) \\ s_2(t) \\ \dots \\ \dots \\ \dots \\ s_{n-2}(t) \\ s_{n-1}(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & \dots & 0 & c_0 \\ 1 & 0 & \dots & 0 & c_1 \\ 0 & 1 & \dots & 0 & c_2 \\ \dots & \dots & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & c_{n-2} \\ 0 & 0 & \dots & 1 & c_{n-1} \end{bmatrix} \oplus \begin{bmatrix} s_0(t-1) \\ s_1(t-1) \\ s_2(t-1) \\ \dots \\ \dots \\ \dots \\ s_{n-2}(t-1) \\ s_{n-1}(t-1) \end{bmatrix}$$

or

$$\boldsymbol{s}(t) = \boldsymbol{F} \oplus \boldsymbol{s}(t-1).$$

The Galois is also called one-to-many or internal XOR gates or modular, since the output is distributed to all clocked cells (called taps). The taps are the clocks that are activated in every iteration, i.e., $c_i = 1$. The diagrammatical representation of the Galois LFSR for n = 5 is shown in Figure 1 below (Robling,,1982).

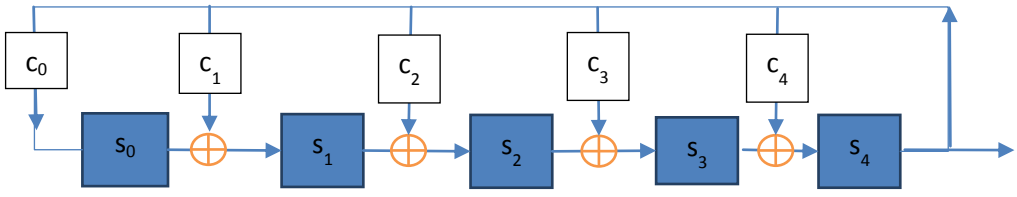

Figure 1: 5-Degree Galois LFSR.

The matrix representation of Figure1 is

$$\begin{pmatrix} s_0(t) \\ s_1(t) \\ s_2(t) \\ s_3(t) \\ s_4(t) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & c_0 \\ 1 & 0 & 0 & 0 & c_1 \\ 0 & 1 & 0 & 0 & c_2 \\ 0 & 0 & 1 & 0 & c_3 \\ 0 & 0 & 0 & 1 & c_4 \end{pmatrix} \oplus \begin{pmatrix} s_0(t-1) \\ s_1(t-1) \\ s_2(t-1) \\ s_3(t-1) \\ s_4(t-1) \end{pmatrix}$$

Example 1: The states for an LFSR whose connection vector is $c = (0,1,0,1,1)$ after 39 iterations are as follows:

S(0) = [1 0 0 0 0]	S(10) = [1 1 1 0 0]	S(20) = [1 0 1 0 1]	S(30) = [1 0 0 0 0]
S(1) = [0 1 0 0 0]	S(11) = [0 1 1 1 0]	S(21) = [1 1 1 1 1]	S(31) = [0 1 0 0 0]
S(2) = [0 0 1 0 0]	S(12) = [0 0 1 1 1]	S(22) = [1 1 0 1 0]	S(32) = [0 0 1 0 0]
S(3) = [0 0 0 1 0]	S(13) = [1 0 1 1 0]	S(23) = [0 1 1 0 1]	S(33) = [0 0 0 1 0]
S(4) = [0 0 0 0 1]	S(14) = [0 1 0 1 1]	S(24) = [1 0 0 1 1]	S(34) = [0 0 0 0 1]
S(5) = [1 0 1 0 1]	S(15) = [1 0 0 0 0]	S(25) = [1 1 1 0 0]	S(35) = [1 0 1 0 1]
S(6) = [1 1 1 1 1]	S(16) = [0 1 0 0 0]	S(26) = [0 1 1 1 0]	S(36) = [1 1 1 1 1]
S(7) = [1 1 0 1 0]	S(17) = [0 0 1 0 0]	S(27) = [0 0 1 1 1]	S(37) = [1 1 0 1 0]
S(8) = [0 1 1 0 1]	S(18) = [0 0 0 1 0]	S(28) = [1 0 1 1 0]	S(38) = [0 1 1 0 1]
S(9) = [1 0 0 1 1]	S(19) = [0 0 0 0 1]	S(29) = [0 1 0 1 1]	S(39) = [1 0 0 1 1]

From this LFSR (Example 1), S(0) = S(15) = S(30). Clearly,

$$S(j) = S(j+15), j = 0,1,2,\ldots$$

We say that the LFSR is period with period 15.

Example 2: The states for an LFSR whose connection vector is $c = (1,0,0,1,0)$ after 39 iterations are as follows:

S(0) = [1 0 0 0 0]	S(2) = [0 0 1 0 0]	S(4) = [0 0 0 0 1]	S(6) = [0 1 0 0 1]
S(1) = [0 1 0 0 0]	S(3) = [0 0 0 1 0]	S(5) = [1 0 0 1 0]	S(7) = [1 0 1 1 0]
S(8) = [0 1 0 1 1]	S(9) = [1 0 1 1 1]	S(10) = [1 1 0 0 1]	S(11) = [1 1 1 1 0]
S(12) = [0 1 1 1 1]	S(13) = [1 0 1 0 1]	S(14) = [1 1 0 0 0]	S(15) = [0 1 1 0 0]
S(16) = [0 0 1 1 0]	S(17) = [0 0 0 1 1]	S(18) = [1 0 0 1 1]	S(19) = [1 1 0 1 1]
S(20) = [1 1 1 1 1]	S(21) = [1 1 1 0 1]	S(22) = [1 1 1 0 0]	S(23) = [0 1 1 1 0]
S(24) = [0 0 1 1 1]	S(25) = [1 0 0 0 1]	S(26) = [1 1 0 1 0]	S(27) = [0 1 1 0 1]
S(28) = [1 0 1 0 0]	S(29) = [0 1 0 1 0]	S(30) = [0 0 1 0 1]	S(31) =[1 0 0 0 0]
S(32) = [0 1 0 0 0]	S(33) = [0 0 1 0 0]	S(34) = [0 0 0 1 0]	S(35)= [0 0 0 0 1]
S(36) = [1 0 0 1 0]	S(37) = [0 1 0 0 1]	S(38) = [1 0 1 1 0]	S(39) = [0 1 0 1 1]

Clearly, $S(j) = S(j+31), j = 0,1,2,\ldots$ This LFSR is periodic with period 31. An LFSR is periodic with period M if M is the lowest integer for which

$$S(0) = S(M)$$

3. MAIN RESULT: USING MATRIX TO COMPUTER MAXIMUM PERIOD LINEAR FEEDBACK SHIFT REGISTERS

An LFSR of length n (length and degree are used interchangeably) whose period is $M = 2^n - 1$ is called a maximum-period LFSR. Note that this implies there are no other integers lower than M for which the LFSR is periodic. Therefore, Example 2 is a maximum-period LFSR while Example 1 is not. Maximum-period LFSRs are of interest in cryptography and in several applications [Benvenuto, 2012;Sachs, 2017]. It is possible to have an LFSR

whose period is a factor of $2^n - 1$ but not equal to $2^n - 1$. In that case, the LFSR has considered a degenerate LFSR. A non-degenerate LFSR is a maximum-period LFSR.

The $s(j)$ is computed using the formula:

$$s(j) = F^j \cdot s(0)\, mod(2).$$

A maximum-period LFSR of length *n* has the following property:

$$s(j) = F^{M+j} \cdot s(0)$$

Our goal is to determine all LFSRs of size *n* that are of maximum period. This is the maximum number of state vectors (not counting zero vector) or the maximum combinations of elements in the connection vector with $c_0 = 1$,as in case of non-degenerate LFSR. A maximum period LFSR is obtained using the formula:

$$s(0) = F^{(2^n-1)} \cdot s(0) = F^1 F^2 F^2 \cdots F^2 F^2 \cdot s(0).$$

For matrix computation, first compute and store in the memory:

$$G_j = F^{2^j}, j = 0,1, \ldots, n-1$$

Then, the F^M is computed using

$$F^M = G_{(n-1)} G_{(n-2)} \cdots G_1 G_0 = F$$

For Example 2, the matrix powers are as follows:

G_0 =[[0 0 0 0 1], [1 0 0 0 0], [0 1 0 0 0], [0 0 1 0 1], [0 0 0 1 0]]
G_1 = [[0 0 1 0 1], [0 0 0 1 0], [0 0 0 0 1], [1 0 1 0 1], [0 1 0 1 0]]
G_2 = [[1 0 1 0 1], [0 1 0 1 0], [0 0 1 0 1], [1 0 1 1 1], [0 1 0 1 1]]
G_3 = [[1 1 1 0 1], [0 1 1 1 0], [1 0 1 1 1], [1 0 1 1 0], [1 1 0 1 1]]
G_4 =[[0 1 1 1 1], [0 0 1 1 1], [0 0 0 1 1], [1 1 1 1 0], [1 1 1 1 1]]
G_4G_3 =[[1 1 0 1 0], [0 1 1 0 1], [0 0 1 1 0], [0 1 0 0 1], [1 0 1 0 0]]
$G_4G_3G_2$ =[[0 0 1 0 0], [1 0 0 1 0], [0 1 0 0 1], [1 0 0 0 0], [0 1 0 0 0]]
$G_4G_3G_2G_1$ =[[1 0 0 00], [0 1 0 0 0], [0 0 1 0 0], [0 0 0 1 0], [0 0 0 0 1]]
$G_4G_3G_2G_1G_0$ =[[0 0 0 0 1], [1 0 0 0 0], [0 1 0 0 0], [0 0 1 0 1], [0 0 0 1 0]]

Using the powers, we only have to compute the G's and use them to determine the F^M. The above formula would work for $M = 2^n - 1$ being Messene, that is, M is a prime number. If M is not Messene, the matrix powers as described will yield LFSR that are not maximum period, an additional criterion is required for he LFSR to be a maximum period and that is, there is no integer less than$M$ that is a factor of $2^n - 1$.

The LFSRs with length of 6 are as follows:

c =[1, 1, 0, 0, 0, 0] has period= 63 and is maximum period
c = [1, 0, 0, 1, 0, 0] has period= 9 and a factor of 63
c = [1, 1, 0, 0, 1, 0] has period= 21 and a factor of 63
c = [1, 1, 1, 0, 1, 0] has period= 21 and a factor of 63
c = [1, 1, 0, 1, 1, 0] has period= 63 and is maximum period
c = [1, 0, 1, 0, 0, 1] has period= 21 and a factor of 63
c = [1, 1, 1, 0, 0, 1] has period= 63 and a maximum period
c = [1, 0, 1, 1, 0, 1] has period= 63 and a maximum period
c = [1, 1, 0, 0, 1, 1] has period= 63 and a maximum period
c = [1, 0, 1, 0, 1, 1] has period= 21 and a factor of 63
c = [1, 1, 1, 1, 1, 1] has period= 7 and a factor of 63

The LSFRs with periods 7, 9, and 21 are not maximum period. However, these values are factors of $2^6 - 1 = 63$.

For these three cases:

$$F^7 = G_2G_1G_0,\ F^9 = G_3G_1,\ F^{21} = G_4G_2G_0$$

The algorithm for determining maximum period is accomplished by starting with the lowest exponent and adding a higher exponent and discarding cases in which the resultant exponent divides $M = 2^n - 1$.

Step 1: e = 1:
Step 2: add the next exponent e := e +e'
Step 3: if M%e != 0, go to Step 2
Step 4: if F^e = F and e < M, LFSR is not maximum period
Step 5: LFSR is maximum period

For lengths 2 to 19, the number of maximum-period LFSRs and the CPU time seconds taken to compute them using the matrix approach are given below. These compute times are comparable to times reported in [Saxena and McCluskey, 2004].

length= 2, # of maximum-period LFSRs = 1, CPU Time = 0.0001201629638671875 secs
length= 3, # of maximum-period LFSRs = 2, CPU Time = 0.00030112266540527344 secs
length= 4, # of maximum-period LFSRs = 2, CPU Time = 0.0006489753723144531 secs
length= 5, # of maximum-period LFSRs = 6, CPU Time = 0.0011839866638183594 secs
length= 6, # of maximum-period LFSRs = 6, CPU Time = 0.002789735794067383 secs
length= 7, # of maximum-period LFSRs = 18, CPU Time = 0.00616002082824707 secs
length= 8, # of maximum-period LFSRs = 16, CPU Time = 0.013325691223144531 secs
length= 9, # of maximum-period LFSRs = 48, CPU Time = 0.031181812286376953 secs
length= 10, # of maximum-period LFSRs = 60, CPU Time = 0.06535077095031738 secs
length= 11, # of maximum-period LFSRs = 176, CPU Time = 0.12902116775512695 secs
length= 12, # of maximum-period LFSRs = 144, CPU Time = 0.28855180740356445 secs
length= 13, # of maximum-period LFSRs = 630, CPU Time = 0.9483609199523926 secs
length= 14, # of maximum-period LFSRs = 756, CPU Time = 1.3931469917297363 secs
length= 15, # of maximum-period LFSRs = 1800, CPU Time = 3.2502479553222656 secs
length= 16, # of maximum-period LFSRs = 2048, CPU Time = 7.728533983230591 secs
length= 17, # of maximum-period LFSRs = 7710, CPU Time = 94.26528286933899 secs
length= 18, # of maximum-period LFSRs = 7776, CPU Time = 39.599074840545654 secs
length= 19, # of maximum-period LFSRs = 27594, CPU Time = 1140.4197480678558 secs

4. CONCLUSION

Linear feedback shift registers are a special type of stream ciphers that are used to efficiently generate pseudo-random sequences. In this paper, we addressed matrix representation of the LFSRs and developed algorithm for determining maximum period LFSRs. Maximum period LFSRs has inherent properties that make them suitable for many security applications, particularly in mobile communications and error correcting systems.

REFERENCES

Benvenuto, C.J. (2012). Galois Field in Cryptography. https://sites.math.washington.edu/~morrow/336_12/papers/juan.pdf

Weisstein, E. W. (undated). "Primitive Polynomial." From *MathWorld*--A Wolfram Web Resource. http://mathworld.wolfram.com/PrimitivePolynomial.html

Bahrami, S. and Majid Naderi, M.(2012). Image Encryption Using a Lightweight Stream Encryption Algorithm. *Advances in Multimedia*.. Article ID 767364, 8 pageshttp://dx.doi.org/10.1155/2012/767364

Paar, C. and Pelzl, J. (2010). Understanding Cryptography, A Textbook for Students and Practitioners. *Springer-Verlag*, Berlin

Robling Denning, D. E. (1982). *Cryptography and data security*. Addison-Wesley Longman Publishing Co., Inc.

Sachs, Jason. (2017). Linear Feedback Shift Registers for the Uninitiated Parts I-XVIII. https://www.embeddedrelated.com/showarticle/1193.php.

Galanis, M., Kitsos, P., Kostopoulos, G., Sklavos, N., and Goutis. C. (2005). Comparison of the Hardware Implementation of Stream Ciphers. The International Arab Journal of Information Technology, Vol. 2, No. 4, pp.267-274

Borowski, M., &Leśniewicz, M. (2012) Modern usage of "old" one-time pad. In *Communications and Information Systems Conference (MCC), 2012 Military* (pp. 1-5). IEEE.

Saxena, N., & McCluskey, E.J. (2004). Primitive Polynomial Generation Algorithms Implementation and Performance Analysis. Stanford University. Center for Reliable Computing. Technical Report No. 04-03.