PES University, Bangalore

(Established under Karnataka Act No. 16 of 2013)

MAY 2020: IN SEMESTER ASSESSMENT (ISA) B.TECH. IV SEMESTER

UE18MA251- LINEAR ALGEBRA

MINI PROJECT REPORT

ON

A Study of Matrix Theory In Shift Register Based Random Number Generation.

Submitted by

1. Name: Madhav Jivrajani SRN: PES1201800028
2. Name: Shriprajwal Krishnamurthy SRN: PES1201800348
3. Name: Ashwin Alaparthi SRN: PES1201802062

Branch & Section : CSE, 4E

PROJECT EVALUATION

(For Official Use Only)

Sl.No.	Parameter	Max Marks	Marks Awarded
1	Background & Framing of the problem	4	
2	Approach and Solution	4	
3	References	4	
4	Clarity of the concepts & Creativity	4	
5	Choice of examples and understanding of the topic	4	
6	Presentation of the work	5	
	Total	25	

Name of the Course Instructor :

Signature of the Course Instructor :

List of Figures

Pg

List of Tables

A Study Of Matrix Theory In Shift Register Based Random Number Generation.

Introduction

Random number generation is a topic that is considered to be of high importance in fields of communication, security, etc. Random numbers can be generated using hardware random number generators (HRNGs) or software random number generators which are also known as pseudo-random number generators. It's called pseudo since no classical random number generator can be *purely* random in nature due to the deterministic nature of the underlying algorithm. Any random number generator can be represented as a Finite State Machine (FSM). As the FSM has only a finite number of states it can be in, the random number generator also has a finite number of states that it can be in before it starts to repeat itself [1]. Therefore, to maximise the apparent lack of pattern in the RNG, it is desirable to maximise the *period* of the RNG. In applications such as stream ciphers, RNGs in the form of shift registers are used and matrix theory and linear algebra provide ways to arrive at RNGs with maximal periods. It is important to know how to arrive at the period of a shift register based RNG and how to configure feedback mechanisms so that the RNG has a maximal period.
A linear feedback shift register (LFSR) is a device that produces a long period pseudo-random bit stream of zeros and ones that is determined by the settings of a relatively small number of switches, and a short initial bit string.

Literature Survey

Introduction to LFSRs:

Linear Feedback Shift Registers (LFSRs) are as the name suggests, shift registers whose input is a linear function of its previous state. They are registers that, when clocked, advance the signal from one bit towards the *Most Significant Bit (MSB).* This is achieved by providing some form of linear feedback from the output to the input and the most common form of

feedback provided is that of an XOR or XNOR gate. Therefore, it is easy to see that the input to the LFSR is a result of the XOR or XNOR of two or more bits of the previous state of the LFSR.

An LFSR consists mainly of two parts:

1. Storage elements (Flip Flops)
2. Feedback path (linear function)

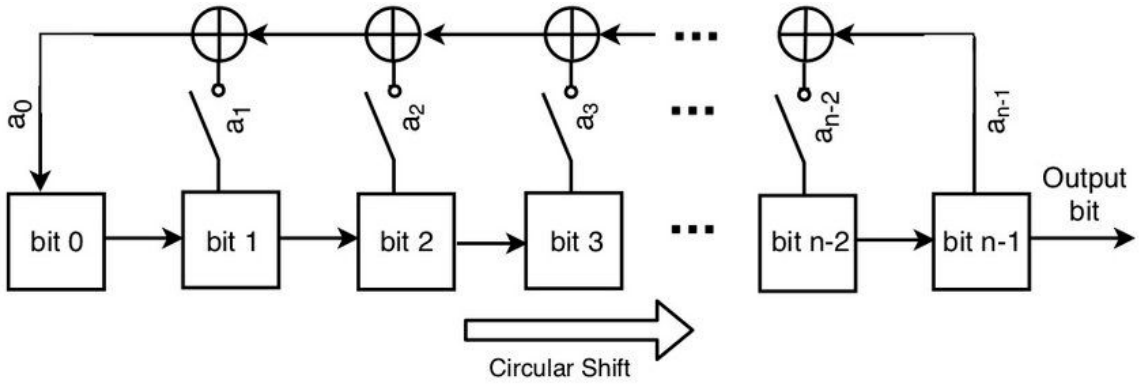


Figure 1: *N-bit LFSR [2]*

Each flip flop has a fixed set of distinct states that it can be in and we have a finite number of flip-flops in our LFSR therefore, the state of the shift register will eventually have to repeat. For all practical purposes, we desire an LFSR that has the longest possible count sequence. This can be achieved by selecting the taps carefully. The maximum count length of an LFSR ideally is equal to 2^m [3] where m is equal to the number of flip-flops in the shift register. As the current state of the register is a linear function of the previous state of the register, the operation performed by the LFSR is purely deterministic in nature i.e. given the state the LFSR is in currently, we can with full certainty, deduce the next state of the LFSR. Even though the operation of an LFSR is deterministic in nature, the output bit sequence produced by an LFSR is pseudo-random in nature. The output bit here refers to the bit coming out of the MSB of the LFSR.

Terminology related to LFSRs [1]:

1. *Degree of an LFSR:* If an LFSR has *m* flip-flops then the degree of the LFSR is said to be *m*.
2. *Lockup state***:** Lockup state is a state in which the next state of the LFSR is the same as the previous state, and as a result of the property that the current state is a linear function of the previous state, every subsequent state will be equal to the lockup state and hence there would be no updation of values that would take place. In case of an XOR feedback mechanism , the lockup state would be a state of all zeros and in the case of an XNOR feedback mechanism the lockup state would be a state of all ones. Therefore, the maximum length of the count sequence for an LFSR with a lockup state would be 2^m - 1, where m is the *degree* of the LFSR.
3. *Taps:* Taps are the position of the bits that are sent as feedback to the XOR or XNOR gates to determine the next state of the shift register. Taps are of very high importance as their positions determine the length of the count sequence of the shift register. If taps are placed in the right positions then the length of the count sequence will be 2^m with no lockup states and 2^m - 1 with a lockup state. where m is the *degree* of the LFSR.
4. *Maximal LFSR:* An LFSR of degree *m* which has a count sequence length of 2^m with no lockup states and 2^m - 1 with a lockup state. It is always desirable to have an LFSR which is a maximal LFSR.
5. *Seed:* Seed or also called Initial Vector is the initial state that the shift register is initialized with. The reason the output bit sequence produced by an LFSR is called pseudo random is that, this sequence depends on what the seed was among a few other factors such as tap positions and feedback function.

Working of LFSR :

An n-stage linear feedback shift register (LFSR) consists of a sequence of n binary storage devices, from F_1 to F_n, each storing either a 0 or 1. At each clock pulse, the value in F_{i+1} is shifted to F_i

The new value of F_n is decided by the feedback, which would be the sum of the values of the previous F_i devices modulo 2

Now, at F_n after the k-th clock cycle, the value a_{n+k} can be found using the recurrence formula [4]-

$$a_{n+k} = \sum_{i=0}^{n-1} c_i a_{k+i}$$

Where c_i is 0 if switch K_{i+1} is open, and 1 if closed.

The values of $\overline{c} = (c_0 , c_1 \ldots . C_n)$ is the coefficient vector and makes up the n bit key. The entries in the initial state vector $\overline{a}$ make up the n bit initial condition.

Finite State Representation :

The state of a LFSR can be represented as $\overline{s}$ where $\overline{s} = (s_1 , s_2 \ldots . s_n)$, and s_i represents the value of the i^{th} register F_n [4]. The transition from $\overline{s}$ to $\overline{s'}$ is given by $\overline{s'}_i = \overline{s}_{i+1}$ when $1 \leq i < n$ and $\overline{s'}_n = \overline{c} \ . \ \overline{s}$

Important terms in matrix theory for LFSRs:

1. *Monic Polynomial:* A monic polynomial is a polynomial whose highest degree coefficient is equal to 1 [5].
2. *Minimal Polynomial:* If T is a matrix belonging to some vector space V then, there exists a unique *monic* polynomial *p* of least degree such that p(T) = 0
3. *Characteristic Polynomial:* Characteristic polynomial of a matrix is a polynomial having the determinant and the trace as coefficients and eigenvalues of the matrix as roots [6]. In most cases the characteristic polynomial is the same as the minimal polynomial. In cases when it is not the same, it is the case that the characteristic polynomial has repeated roots as might be the case with a triangular matrix.
4. *Field:* A field is a set F together with two binary operations on F called addition and multiplication. A binary operation on F is a mapping $F \times F \rightarrow F$, that is, a correspondence that associates with each ordered pair of elements of F a uniquely determined element of F [7]. More specifically, when this set is finite in nature, this field is known as a *finite field* or a *Galois field.*
5. *GF(2):* GF(2) stands for the galois field of two elements: {0, 1}, it is the smallest field. GF(2) is of high importance in the theory of LFSRs as the feedback polynomial

is a polynomial from the GF(2) field, which means that the coefficients of the terms in the polynomial are always 0 or 1 [1]. The general form of a generator polynomial in the GF(2) field is:

$$G(X) = g_m X^m + g_{m-1} X^{m-1} \ldots g_1 X + g_0$$

The coefficients g_i represent the tap weights and can take values of 0 or 1. The value of 1 indicates that there is a feedback tap from the state *i*. All mathematical operations are performed in modulo 2. g_m is 1 in the case of G(X) being a minimal polynomial.

6. *Companion Matrix:* A companion matrix is defined for a *monic polynomial.* Consider the monic polynomial [8]

$$p(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_{n-1} x^{n-1} + x^n$$

The companion matrix for the polynomial p is

$$C(p) = \begin{bmatrix} 0 & 0 & \ldots & 0 & -c_0 \\ 1 & 0 & \ldots & 0 & -c_1 \\ 0 & 1 & \ldots & 0 & -c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & -c_{n-1} \end{bmatrix}$$

For the purposes of linear feedback recurrences as in the case of LFSRs, the transpose of the companion matrix is used to act as a generator.

7. *Reciprocal Polynomial:* The reciprocal polynomial of a polynomial *p* is a polynomial in which the coefficients are written in reverse order. The concept of reciprocal polynomials is often seen in the case of LFSRs. If a matrix *M* has a characteristic polynomial *p,* the reciprocal polynomial of *p* is the characteristic polynomial of the inverse of *M.*
8. *Feedback Polynomial:* The reciprocal polynomial of the characteristic polynomial of the LFSR is called the feedback polynomial of the LFSR [1].

Matrix Model :

The "shift" in shift registers is modeled as a linear transformation from state *s* to state *s'* [8]. Each sequence of an n stage LFSR can be looked at as an *n-dimensional* vector. More formally, a left shift operator L can be defined as follows:

For any sequence

$$\{a_i\}_{i\geq 0} = (a_0, a_1, a_2, ...) \in V(F_q),$$
$$L(a_0, a_1, a_2, ...) = (a_1, a_2, ...)$$

A sequence of only zeros is transformed to itself. Therefore, if an LFSR with an XOR based feedback encounters an all zero state then it is said to be degenerate and it will never come out of this state.

The transition from s to s' is given by $s'_n = c_0 s_1 + ... + c_{n-1} s_n$

This is given in the matrix model by $\overline{s'} = \overline{s}\,\mathrm{M}$, where M is a n x n matrix with

$$m_{ij} = \begin{cases} 1, & \text{if } i = j+1 \\ c_{i-1} & \text{if } j = n \\ 0 & \text{otherwise} \end{cases}$$

If c_0 is equal to 0 then the matrix *M* is singular. This state of the LFSR is known as the degenerate state of the LFSR and in this state, a *n* stage LFSR becomes equivalent to an *n-1* stage LFSR [4]. Therefore, it's c_0 is always considered to be 1 and *M* is non-singular in nature. Thus, a mapping from the current state of the LFSR to the next state of the LFSR can be established and the next state of the LFSR is a permutation of the one of the 2^n binary state vectors.

Theorem 1: Let *T* be a linear operator on an n-dimensional vector space V [9]. The characteristic and minimal polynomials for *T* have the same roots except for the multiplicities.

Proof: Let p be the minimal polynomial for T. Let c be a scalar. What we want to show is that p(c) = 0 if and only if c is an eigenvalue of T. Then, we have

$$p = (x - c)q$$

where *q* is a polynomial. Since deg(q) < deg(p), the definition of the minimal polynomials *p* tells us that $q(T) \neq 0$. Choose a vector *b* such that $q(T)b \neq 0$. Let $a = q(T)b$. Then

$$0 = p(T)b$$
$$= (T - cI)q(T)b$$
$$= (T - cI)a$$

This shows that *c* is an eigenvalue of *T*. Hence, assuming $Ta = ca$ with *a* not equal to 0, we can say that

$$p(T)a = p(c)a$$

Since, $p(T) = 0$ and *a* not equal to 0, we have $p(c) = 0$

Theorem 2: Let *M* be a nonsingular matrix over a finite field K with minimum polynomial *m(x).* Then the period of *M* is the smallest positive integer such that *m(x)* divides $x^p - 1$.

Proof: Staying in the scope of the GF(2) field, there are a finite number of *n* x *n* matrices over the two-element field GF(2) = {0, 1} of integers modulo 2. Hence, it can be guaranteed [4] that there exist two distinct integers *s* and *t*, such that $0 \leq s < t$ and $M^s = M^t$ Since *M* is nonsingular, we conclude that M is invertible, and hence there must be an integer *h* such that $M^h = I$, *I* being the identity matrix for $h = t - s$. The matrix defined by the above-mentioned equation is the *companion matrix* M = C(m) where m is the polynomial [9]

$$m(x) = x^n - c_{n-1}x^{n-1} - \ldots - c_1x - c_0$$

We also know that if *C(x)* is the companion matrix of a monic polynomial *m(x)* then *m(x)* is both the characteristic and minimal polynomial of *C(x)* or in other words, no roots have their multiplicities exactly equal to 1 which means that all the eigenvectors of *C(x)* are *linearly independent.* From the definition of minimal polynomial, we know that m(M) = 0, and that $f(M) = 0$ if and only if *m(x)* divides *f(x)* or $M^k - I = 0$ if and only if *m(x)* divides $x^k - 1$.

Theorem 3: There exists a monic polynomial p of minimum degree such that p(T) is 0, where T is an operator acting on the vector space V, and $T \in \zeta(V)$.

Proof:

Let,

$$n = dim(v),\ k = n^2$$

Then there is a unique polynomial p of smallest degree such that:

$$p(T) = 0$$

Then the list, 1, T, T^2… T^k is not linearly independent as $\zeta(V)$ has dimension k, and the length of this list is k+1.

Let m be the minimum possible integer such that 1, T, T^2… T^m is linearly dependent. The Linear Dependence Lemma implies that one T^m is a linear combination of all the other elements in the list. It has to T^m because otherwise, we could have chosen a smaller value of m.

Thus, there exists scalars a_0, a_1… a_{m-1} such that:

$$a_0 I + a_1 T + a_2 T^2 + ... + a_{m-1} T^{m-1} + T^m = 0$$

Let us define a monic polynomial p such that:

$$p(z) = a_0 + a_1 z + a_2 z^2 + ... + a_{m-1} z^{m-1} + z^m$$

Therefore, by construction we get:

$$p(T) = 0$$

There cannot be a monic polynomial q that satisfies this condition, that has a degree less than m because of the definition of m [5], that is, it is the minimum possible integer such that the list is not linearly independent.

The polynomial q, cannot be of degree m either as:

$$p(T) = 0$$

$$q(T) = 0$$

$$(p - q)(T) = 0$$

The degree of (p-q) is less than m, as both p and q are monic. Thus upon subtracting them, we lose the term that was raised to the m^{th} power. The maximum possible degree of (p-q) is thus m-1. But we have already proven that there cannot be a polynomial that satisfies the same conditions as p and have degree less than m. Thus, q=p.

Therefore, the existence of the minimal polynomial has been verified.

Theorem 4: If there is a polynomial q such that q(T) = 0. This implies that the polynomial q must be a multiple of the minimal polynomial.

Proof: Let p be the minimal polynomial and q be one of its multiples. This means that there is another polynomial s, such that:

$$q = ps$$

Thus we have:

$$q(T) = p(T)s(T) = 0$$

Thus we now know that q(T) = 0 if it is a multiple of the minimal polynomial.
Suppose q'(T) is a polynomial such that $q'(T) = 0$

By using the division algorithm for the division of polynomials, q'(T) must be of the form

$$q'(T) = p(T)s(T) + r(T)$$

But we know that, since p is the minimal polynomial, p(T) = 0. Thus the above equation reduces to:

$$q'(T) = r(T) = 0$$

Thus, we get that r(T) is zero.

This means that q'(T) is of the form:

$$q'(T) = p(T)s(T) = 0$$

Therefore, q'(T) is a multiple of the minimal polynomial if it is equal to 0.
We have now completed the proof of this theorem.

Corollary: The characteristic polynomial must be a multiple of the minimal polynomial. The Cayley-Hamilton Theorem states that when the characteristic polynomial is applied to T, we get 0 [5]. Thus, upon replacing q(T) with the characteristic polynomial we can easily conclude that the characteristic polynomial is indeed the minimal polynomial.

Maximal Periods:

The *exponent* of a polynomial *f(x)* over a field *K* is defined to be the smallest positive integer *k* such that *f(x)* divides $x^k - 1$ or 0 if no such *k* exists. Theorem 2 tells us that the period of a non-singular matrix over a finite field K is the same as the exponent of its minimal polynomial. If such a matrix has a positive period then it establishes the fact that if *f(x)* is a monic polynomial and $f(0) \neq 0$ then *f* has a positive exponent.

Let $a_0,\ a_1, \ldots a_n$ be the initial seed of the LFSR. If q is the least positive integer such that $a_{k+q} = a_k$ for all positive integers k (let q be the period) . We define,

$$a(k) = (a_k,\ a_{k+1}, \ldots,\ a_{k+n-1})$$

Knowing the properties of the matrix M

$$a(k) = a(k-1)M = aM^k$$

$a(0)$ is nothing but the seed of the LFSR and the bitstream (a_k) has a period q if and only if q is the smallest positive integer such that $aM^q = a$ however, this does not necessarily imply that $M^q = I$ however, in this case it does imply that $M^q - I$ is singular in nature and particularly in this case, it can be said that $M^q = I$ since $aM^q = a$ for every non-zero binary vector a which implies the null space of $M^q - I$ is of dimension n.

Theorem 5: Let p be the period of a *nondegenerate* n-stage LFSR whose matrix is M. The bit sequence with initial vector $a = (1, 0, \ldots,\ 0)$ has a period of p where $n \le p < 2^n$ and every sub-period q will divide p. If p is maximum then p is the only period of the LFSR.

Proof: Let a be a bit stream whose period is s. Then from *theorem 2* we have say that s is the smallest positive integer such that $aM^s = a$ and

$$a(k)M^s = aM^{s+k} = aM^k = a(k)$$

for every positive integer k. Thus, we can say that M^s acts as an identity on the vectors $a(1),\ a(2),\ \ldots,\ a(n)$. But we have $a(1) = aM = (0,\ 0,\ 0, \ldots, 0, 1)$ and $a(k) = (a_k,\ a_{k+1},\ \ldots,\ a_{k+n-1})$ begins with $n - k$ number of zeros followed by a one in position $n - k + 1$ [4].

Therefore, if K is a field over *GF(2)* [1] then the set

$$\{a(1),\ a(2), \ldots,\ a(n)\}$$

will form a basis for the vector space K^n of all n-tuples. Since, M^s acts as an identity on the basis of a vector space, then it must be the identity itself. Hence, s is the smallest positive integer such that $M^s = I$ that is, $s = p$ is the period of M.

If p is the period of M then p is also a subperiod and we know that $p \le 2^n - 1$. Since, $a(p + 1) = a(1)$ and $\{a(1),\ a(2),\ \ldots,\ a(n)\}$ is independent, from *Theorem 1* it follows that $n \le p$.

Suppose that q is a subperiod of M. Since, q is a subperiod there exists a nonzero vector v such that $vM^q = v$. Let $p = dq + r$ with $0 \le r < q$. We also know that $M^p = I$ and therefore, we have $v = vM^p = vM^{dq+r} = v(M^q)^d M^r = vM^r$. Based on the constraints of r, we conclude that $r = 0$ which implies q divides p.

Assuming p is the period of M and p is maximum. Then, $a(k) = aM^k$ for $0 \le k < 2^n - 1$ generates all $2^n - 1$ vectors in K^n and each of these vectors has a period of p. Therefore, the only subperiod is $q = p = 2^n - 1$

Design and Implementation

The goal was to implement an image encryption application using random numbers generated by the LFSR. The design was attempted to be made such that the user would have a choice between two different types of LFSRs: (i) Fibonacci LFSR and (ii) Galois LFSR.

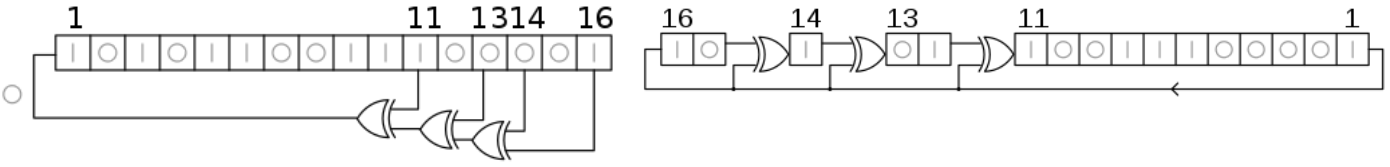


Figure 2: *Fibonacci and Galois LFSR*

Notice that the order of taps in a Fibonacci LFSR is opposite to that of Galois.

The user decides on a binary key which can be generated at random or using principles of private cryptography and key distribution schemes. This acts as a seed to the LFSR. This binary key acts as a password that a user who has encrypted data can use to decrypt it and get back the data. After a binary key haThe next step is to construct a companion matrix that ensures that the LFSR is maximal in nature by choosing pre-calculated characteristic polynomials whose companion matrices are guaranteed to have maximal period.

From the generated companion matrix, we can generate all possible $2^n - 1$ binary vectors of dimension n by performing the following [10]:

$$\begin{pmatrix} a_k \\ a_{k+1} \\ a_{k+2} \\ \vdots \\ a_{k+n-1} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_0 & c_1 & c_2 & \cdots & c_{n-1} \end{pmatrix} \begin{pmatrix} a_{k-1} \\ a_k \\ a_{k+1} \\ \vdots \\ a_{k+n-2} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_0 & c_1 & c_2 & \cdots & c_{n-1} \end{pmatrix}^k \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

Figure 3: *Matrix relation for Fibonacci LFSR*

$$\begin{pmatrix} a_k \\ a_{k+1} \\ a_{k+2} \\ \vdots \\ a_{k+n-1} \end{pmatrix} = \begin{pmatrix} c_0 & 1 & 0 & \cdots & 0 \\ c_1 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{n-1} & 0 & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} a_{k-1} \\ a_k \\ a_{k+1} \\ \vdots \\ a_{k+n-2} \end{pmatrix} = \begin{pmatrix} c_0 & 1 & 0 & \cdots & 0 \\ c_1 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{n-1} & 0 & 0 & \cdots & 0 \end{pmatrix}^k \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

Figure 4: *Matrix relation for Galois LFSR*

After all possible binary vectors have been generated, each binary vector is converted to its decimal equivalent and a set of randomly generated integers are returned. Next, we take each pixel of an image and XOR each of its Red, Green and Blue components with a different randomly generated integer. We do this in a cyclic manner, so values are bound to be repeated which is why it is advisable to use an LFSR of long lengths to achieve a higher degree of randomness or rather to reduce repetition of values that are used for XOR.

Due to the nature of the XOR gate, a user with an encrypted image wishing to decrypt it can use the same LFSR to decode it provided he/she has the same key that the sender used to encrypt the image.

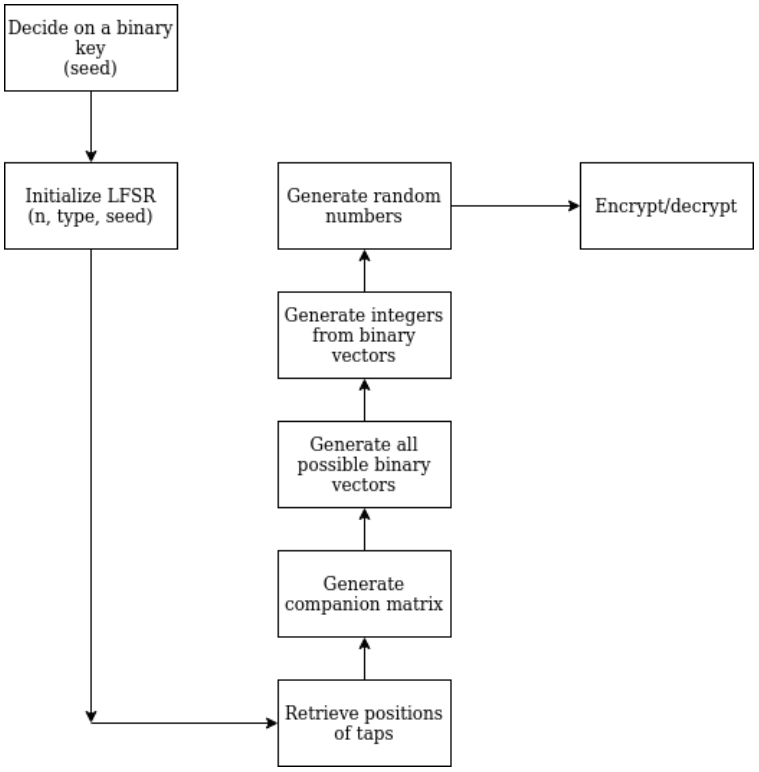


Figure 5: *Design flow*

Example:
Length of LFSR = 15,
Type = Galois
Key (seed) = 111111100100110

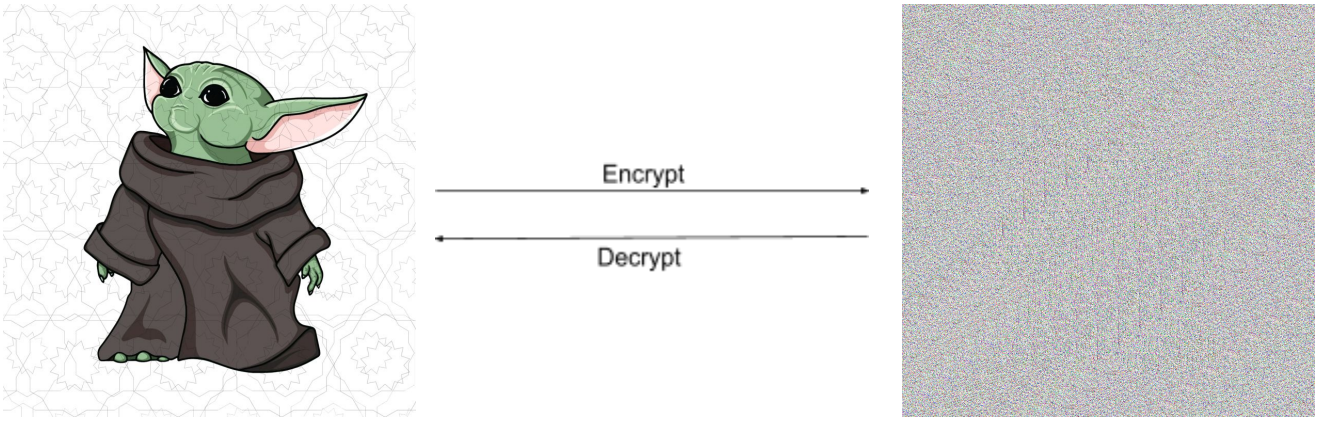


Figure 6: *Example of encryption and decryption*

Another reason to use LFSRs of long length is to make it exponentially difficult to brute force guessing the values with which the pixel values were XORed. For an LFSR of length 15 as above, there are $2^{15} - 1$ distinct values, and this number will only grow exponentially as the length of the LFSR increases.

Applications

We can use random numbers for a wide variety of applications. Some of them are listed below:

1. *Simulation*: Random numbers are generally necessary to make a realistic model of any natural phenomena. These simulations include economic, traffic, nuclear physics, n-body simulations and many other models. Monte-Carlo methods in particular, have a heavy reliance on the quality of the RNG used in their working.

2. *Cryptography*: Public key cryptography systems make use of large amounts of random data. Cryptography relies on deterministic random number generators to recreate a number, unique to the seed provided. Thus, they rely on pseudorandom number generators, where the sender and receiver have the same key used to encode and decode the text sent. This is used in networks to transfer data securely. For example, RSA requires the use of large, random primes for its security. One-time pads require a long keystream of random integers (if these integers are generated by a periodic pRNG [10], the period needs to be longer than the message to be encoded, or the cipher effectively degenerates into a Vigenere cipher, which is vulnerable to attack (unlike a one-time pad))

3. *Statistical Sampling*: Random sampling is one of the most relied upon methods in statistics. As the name suggests, random sampling works only as good as the pRNG used to perform the sampling. If there is some bias in the RNG used to sample, then the sample chosen could also show bias.

4. *Programming*: We can get a rough measure of how long the programs take to run and their algorithmic complexity by feeding the program inputs of random sizes. If the random

input fed to the program has some underlying bias, we might get misleading results. Random numbers are also used in a variety of other algorithms.

5 *Games*: Games use random number generators in a lot of different ways. Games like pokemon use random number generators right from how much damage a move does, to will the player encounter a pokemon at an instant.

Results and Discussions

There are a variety of tests that we can use for testing the statistical properties of pseudo-random number generators.

1. *Generating a bitmap image of the output stream*: We can take the output stream of the random number generator and make each bit in the output stream represent an image. If the entry in the output stream is 0, then it represents a white pixel and if it is 1, then it represents a black pixel. The pseudo random number generator is a bad one if it shows any obvious visual signs of order. The image should resemble noise if the output stream is close to a totally random bit stream.

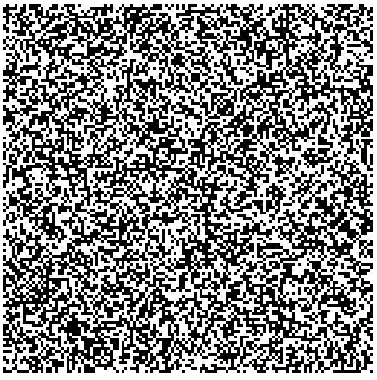


Figure 7: Bitmap of the LFSR output stream

2. *Series Test:* In this test, we count the number of occurrences of all permutations of various n-bit strings in the output stream. In Table 1, we calculated the number of occurrences of 2-bit strings(that is, 00, 01, 10, 11) and 3 bit strings in the output stream of the LFSR

Pattern	Frequency in the output stream
00	2730
01	4096
10	4096
11	2731
000	1170
001	2048
010	1638
011	2048
100	2048
101	1639
110	2047
111	1169

Table 1: Results of the Series test on the output stream of LFSR

3. *Run test:* A "run" is simply a string of bits that occur without transitioning. For example: 110100 has 4 runs: 11, 0, 1 and 00. The length of the run is the number of elements in it [11]. Exactly half of these runs are one bit long, a quarter are two bits long, up to a single run of zeros $n - 1$ bits long, and a single run of ones n bits long. This distribution almost equals the statistical expectation value for a truly random sequence. However, the probability of finding exactly this distribution in a sample of a truly random sequence is rather low

Length of the run	Frequency of such a run
1	4097
2	2048
3	1023
4	512
5	256
6	128
7	64
8	32
9	16
10	8
11	4
12	2
13	1
14	1

Table 2: Results of the run test on the output stream of the LFSR

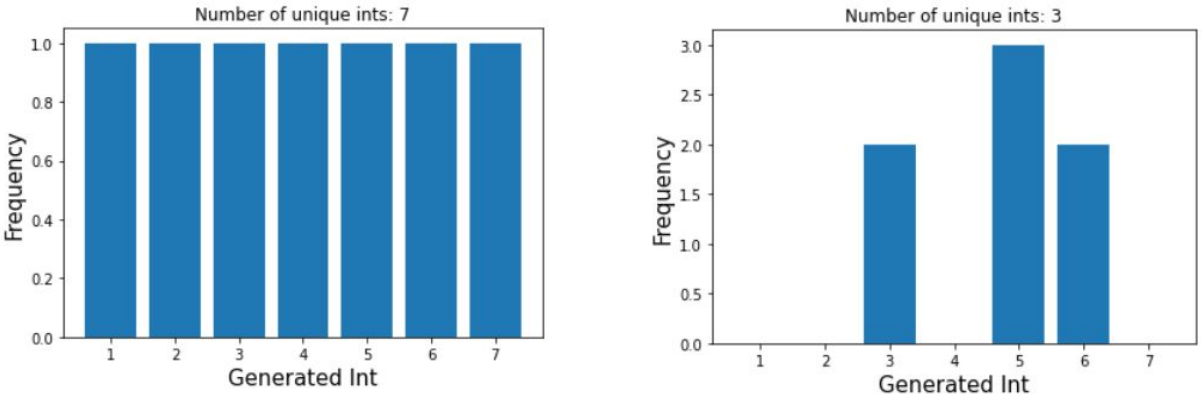
Conclusion

As discussed, the need for random number generators that have substantially long periods is important. Linear algebra plays an undeniably essential role in deciding what the configuration of taps should be in order to construct LFSRs which are maximal in nature. With the help of matrix theory and field theory, we can derive and construct even better random number generators by combining two random number generators.

Here's an example to demonstrate the importance of having LFSRs that are maximal in nature.

$$C_0(x) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} \qquad C_1(x) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Figure 8: *C_0 gives maximal LFSR, C_1 does not give maximal LFSR*



(a) (b)

Figure 9: *(a) Maximal LFSR, (b) non-maximal LFSR with short period and repeated integers*

References

[1] S. Hassan and M.U Bokhari, *Design of Pseudo Random Number Generator using Linear Feedback Shift Register.*

[2] S.K. Monfared, O Hajihassani, S M Zanjani, M Kiarostami, D Rahmati, S Gorgin 2019. *Generating High Quality Random Numbers: A High Throughput Parallel Bitsliced Approach.*

[3] P. Alfke, *An Application Note on Efficient Shift Registers, LFSRCounters, and Long Pseudo-Random Sequence Generators,* Xilinx.

[4] W.P. Wardlaw, *A Matrix Model for the Linear Feedback Shift Register.*

[5] S. Axler, *Linear Algebra Done Right,* Third Edition, Chapter 9, Pages 275-285, Springer International Publishing, New York, 2015

[6] G. Strang, *Linear Algebra and its Applications,* Fifth Edition, Chapter 8-11, Page 401-524, Wellesley-Cambridge Press, 2016

[7] K. H. Rosen, *Discrete Mathematics and Its Applications,* Seventh Edition, Chapter 11,

Page 753-760, McGraw Hill Education (India) Private Limited, 2011.

[8] Marsaglia, G. and Tsay, L.H 1984. *Matrices and the Structure of Random Number Sequences.*

[9] K. Hoffman and R. Kunze, *Linear Algebra,* Second Edition, Chapter 6, Page 190-210, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[10] D. Biebighauser, *Testing Random Number Generators,* University of Minnesota.

[11] J.D. Parker, *The Period of The Fibonacci Random Number Generator.*