International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-2, December, 2019

Design of Pseudo Random Number Generator using Linear Feedback Shift Register

Shabbir Hassan, M. U. Bokhari

***Abstract:** - Nowadays security has become a great concern in the field of computer science and information technology. In order to protect data from unintended users and to achieve a desirable level of security, several cryptographic algorithms based on various technology have been proposed. Linear Feedback Shift Register (LFSR) may play an important role in the design of such cryptographic algorithms. LFSR based cryptographic algorithms are often lightweight in nature and are more suitable for resource constraining devices. In this paper we present a detailed analysis of LFSR and design of $m - sequence$ LFSR to implement cryptographic algorithms.*

***Keywords:** Q Array PRNG, FSR, Modular Arithmetic, Galois Field $GF(p^m)$, Primitive Polynomial $p(x)$, Primitive Polynomial $p(x)$ over $GF(p^m)$, LFSR, $m - sequence$, Run Length, Linear Recurrence, NIST.*

I. INTRODUCTION

The paper is written to represent an introductory into the depth concepts of Linear Feedback Shift Registers, Primitive Polynomial and Galois Field. Since these branches of mathematics have a major contribution in computer science and applications. In order to secure data in a communication channel we use several encryption algorithms, keys and mathematical tools. All these attributes can be easily implemented using mathematical concept that has been dealt in this paper. Modular arithmetic and equivalence classes are widely used to implement a cryptosystem, this paper is written to insight all these mathematical tools with a suitable example. Due to the simple structure and implementation of LFSR, it is widely used in network communication and industries for generating pseudo random sequences. LFSR has the widest implementation in networking and cryptography. Generally LFSRs are constructed by D Flip-Flop and two input XOR gates. It can be implemented in two ways, first is Fibonacci implementation and other is D Flip-Flop implementation. An LFSR with maximal length sequence output is called $m - sequence$ LFSR and is widely used in experimental design to get magnetic resonance imaging experiments, a pioneering non-invasive technology for studying the behavior of human brain and to generate Pseudo Noise (PN) or a Pseudorandom Sequences (PS). the functional magnetic resonance imaging experiment shows that the brain stimuli have correlated with the magnetic resonance scanner of the brain to collect functional MRI data for statistical analysis. An $m - sequence$ can also be used to justify the order appearance and timing of the brain stimuli. Due to this promising work, primitive polynomial and $m - sequence$ LFSRs have gained more acceptance in practice. In this paper we present modular arithmetic, primitive polynomial over Galois Field, LFSR and statistical inference of $m - sequence$ LFSR along with their related attributes.

Revised Manuscript Received on December 30, 2019.

* Correspondence Author
Shabbir Hassan, Department of Computer Science, Aligarh Muslim University Aligarh 202002, India
Prof. M. U. Bokhari, Department of Computer Science, Aligarh Muslim University Aligarh 202002, India

II. MOTIVATION

Leonardo Bonacci known as Fibonacci was an Italian mathematician is considered being the most talented western mathematician of the middle age. The Fibonacci numbers are the sequence of numbers defined by a linear recurrence equation:

$$s_n = s_{n-1} + s_{n-2}$$

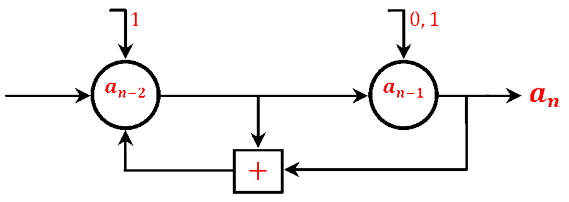


Figure. I

In this series the next number is obtained by adding the previous two numbers. By convenience the first two term are either (0 or 1) or (1 and 1). Fibonacci numbers are closely related to the Lucas numbers. A Fibonacci generator is shown in Figure I. Fibonacci numbers are widely used in computer science to develop algorithms, Fibonacci Search and to implement Fibonacci Heap Data Structure. It is also used to interconnect Distributed System themselves. In the field of biology, Fibonacci number play a vital role in tree branching, arranging leaves on a tree's stem (Phyllotaxis), bearing pineapple sprouts, producing uncurling fern and many more. These application of Fibonacci number motivate us to gather some idea and to correlate Fibonacci series with LFSR.

III.MODULAR ARITHMETIC

The concept of congruence was first introduced by the German mathematician "*Karl Friedrich Gauss*". Let m be a fixed integer, then an integer a is said to be congruent to another integer $r\ modulo(m)$ if $m|(a - r)$ and is denoted as $a \equiv r\ modulo(m)$.

Where m is called modulus and r is called residue. If $0 \leq r \leq m$ then r is called *Least Residue* where as if $0 \leq |r| \leq \frac{m}{2}$ then r is called *Minimal Residue*. For some variable $a = 13$ and $m = 9$, the value of residue r could be $4, -5$ and many more. Computation of the residue is a very typical work in applied cryptography. For the better consideration let us take a nice example. $\forall a, r \in \mathbb{Z}$, assume that $a = 42$ and $m = 9$, few possible values for r may be $-3, -12, 6, 15 \ldots$ and many more. Hence it ensures that for residue r infinite many unique residues are possible. Now let's try to compute the residue of $13 \cdot 16 - 8$ under $modulo(5)$ operation, that is what would be the remainder when $13 \cdot 16 - 8$ is divided by 5. After doing simple arithmetic, obviously the residue will be zero, similarly $5^{48} modulo(24)$ would leave residue 1.

However the above approach for finding residue is too difficult. For convenience a lucid approach had developed is called *Equivalence Classes*. An equivalence class is nothing but it is a set of all possible residue. Let's assume any integer $a = 12$ and $m = 5$, the residue r may be $2, 7, -8, 17$ and many more. So the possible presumed set of elements that can be a value of residue r is an equivalence class of $modulo(5)$. In this case $\{\ldots -13, -8, 2, 7, 12, 17 \ldots\}$ form an equivalence class with $modulo(5)$. All the elements of the class behave equivalent under $modulo(5)$ operation. Let's see what trick plays when we compute the residue of a large value in equivalence classes. There exist N distinct equivalence classes of $modulo(N)$.

$$\begin{array}{rcr|c|l}
A & = & \{\ \cdots -15, -10, -5, & \mathbf{0}, & 5, 10, 15, \cdots\ \} \\
B & = & \{\ \cdots -14, -9, -4, & \mathbf{1}, & 6, 11, 16, \cdots\ \} \\
C & = & \{\ \cdots\ -1,\ -8, -3, & \mathbf{2}, & 7, 12, 17, \cdots\ \} \\
D & = & \{\ \cdots -12, -7, -2, & \mathbf{3}, & 8, 13, 18, \cdots\ \} \\
E & = & \{\ \cdots -11, -6, -1, & \mathbf{4}, & 9, 14, 19, \cdots\ \}
\end{array}$$

Now let's try to compute the residue of $(13 \cdot 16 - 8) modulo(5)$ with the help of $modulo(5)$ equivalence classes. In this approach the numeric value of problem is replaced by the name of class for which they belongs, then reduced the alphabetic expression if possible, and at the end we substitute any of the least values corresponding to that particular class. Since any of the equivalence classes are not a set of integers or real numbers itself. It can be seem that it has very less number of elements as of $\mathbb{Z}$ *or* $\mathbb{R}$. Obviously it will take a small complexity to search an element as compared with the complexity required in $\mathbb{Z}$ *or* $\mathbb{R}$. e.g. to find the value of $(13 \cdot 16 - 8)\, modulo(5)$, we must substitute character D instead of 13 and 8, and B instead of 16 because13 and 8 $\in D$ and 16 $\in B$.

$$\boldsymbol{(D \cdot B - D) modulo(5) \approx D(B - 1) modulo(5)}$$

Now substitute any probably small elements belonging to the corresponding class B and D say 1 and 3. Actually 1 is the $modulo(5)$ of 16, while 3 is the $modulo(5)$ of both 8 and 18 so;

$$\boldsymbol{3(1 - 1) modulo(5) = 0 modulo(5)}$$

Hence again the residue is 0, that has been obtained by doing the actual paper pencil calculation. In public key cryptography's most asymmetric cryptosystem are based on modular arithmetic to compute the residue for a large exponent, these are computed in our web browser when we make a secure connection with eBay, Amazon, Flip Kart etc. by running the HTTPS protocol background. But in such cases the number are approx. 2000−bit long.

IV. GALOIS FIELD

Galois Field named after "*Evariste Galois*", also known as Finite Field, refer to a field in which there exist finitely many element. It is particularly useful in translating computer data as they are represented in the binary vectors. Since the elements of vector are the member of the finite set $S = \{0, 1\}$, that are the element of Galois Field having 2 element, also called Prime Field as shown in the Figure II. The Advance Encryption Standard (AES) utilizes the ideas of Galois Field. A Galois Field exist if and only if, it has p^melements, where $p \in \mathbb{P}\ and\ m \in \mathbb{Z}^+, p$ is the ***characteristic*** of the Field, however order of the Field p^m represents the number of elements it contains. For instance, a finite Field with 11 elements is $GF(11)$, and with 81 elements are $GF(81)\ or\ GF(3^4)$, however $GF(2^8)$ represent a Finite Field with 256 elements. Concept of Galois Field is used in our web browser to establish a secure connection on HTTPS. LFSR perform its multiplication on Galois Field. The elements of the Galois Field $GF(p^m)$ is defined as:

$$\begin{aligned}
GF(p^m) = \ & (0, 1, 2, \ldots, (p-1)) \cup \\
& p, (p+1), (p+2), \ldots, (p+p-1) \cup \\
& p^2, (p^2+1), (p^2+2), \ldots, (p^2+p-1) \\
& \qquad\qquad \cup \\
& \qquad\qquad \vdots \\
& p^{m-1}, (p^{m-1}+1), (p^{m-1}+2), \ldots, (p^{m-1} \\
& \qquad\qquad + p - 1)
\end{aligned}$$

The order of the Field is given by p^m while p is called the characteristic of the Field. From the above generalization, we can say that a Galois Field $GF(5)$ must have $(0, 1, 2, 3, 4)$ elements in it, where each element represent a polynomial of degree zero. While the Galois Field $GF(2^3) = (0, 1, 2, (2+1), 2^2, (2^2+1), (2^2+2), (2^2 + 2 + 1))$ yields element $(0, 1, 2, 3, 4, 5, 6, 7)$, each of these element represents a polynomial of degree at most two [2, 3]. Concept of Galois Field is widely used in the field of Cryptography. Since each byte represents as a vector of a Finite Field, encryption & decryption using mathematical arithmetic are very easy [6].

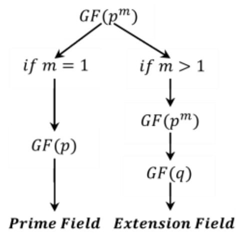


Figure II. PRNG Sequence

V. Construction of a Galois Field $GF(p^m)$

An extended Galois Field contain not only the elements from its "*Ground Field*" $GF(p)$, but also an element α such that $ord(\alpha) = p^m - 1$ and the $p^m - 1$ consecutive power of α i.e. $\{\alpha^0, \alpha^1, \dots, \alpha^{ord(\alpha)-1}\}$ are the nonzero elements of the Extended Field $GF(p^m)$ or $GF(q)$ and hold $ord(\alpha^i) = \frac{p^m-1}{(p^m-1, i)}$ [1]. Now pretend that the set contains infinite many elements, as a set these elements may be represented as:

$$\boldsymbol{GF(q) = \{0, 1, \alpha, \alpha^2, \dots, \alpha^k, \dots .\}}$$

So an element α^i is nothing but a root of primitive polynomial $p(x) \in GF(2)[x]$ of degree m, therefore $p(\alpha) = 0$ and also $p(x)|x^d - 1$ where $d = 2^m - 1$ therefore:

$$\boldsymbol{x^d - 1 = p(x) \cdot q(x)}$$

$$\boldsymbol{\alpha^d - 1 = p(\alpha) \cdot q(\alpha) = 0}$$

$\alpha^d = 1$, so again it generates the same element 1, hence our postulate becomes false, and we ensure that $GF(q)$ is a Finite Field that contains 2^d and a zero that is 2^m elements. Therefore the set is a Field of 2^m has finite elements and is of the form of $GF(q) = \{0, 1, \alpha, \alpha^2, \dots, \alpha^{d-1}\}$.

TABLE I

Generation of Elements of $GF(2^4)$

α^0	=	1	α^8	=	$x^2 + 1$
α^1	=	x	α^9	=	$x^3 + x$
α^2	=	x^2	α^{10}	=	$x^2 + x + 1$
α^3	=	x^3	α^{11}	=	$x^3 + x^2 + x$
α^4	=	$x + 1$	α^{12}	=	$x^3 + x^2 + x + 1$
α^5	=	$x^2 + x$	α^{13}	=	$x^3 + x^2 + 1$
α^6	=	$x^3 + x^2$	α^{14}	=	$x^3 + 1$
α^7	=	$x^3 + x + 1$	α^{15}	=	1

For instance let us assume that, $p(x) = x^4 + x + 1$ is a primitive polynomial over $GF(p^m)$ with $m = 4$ and $p = 2$, if α is a primitive element of $p(x)$ so the successive power of α will generate all the non$-$zero elements of the $GF(16)$ as shown in Table I [1]. An element α^i in the Field is generated by mounting the power of αunder$modulo\, p(x)$. For example the element α^8 is generated as:

$$\boldsymbol{\alpha^8 modulo(x^4 + x + 1)}$$

$$\boldsymbol{x^8 modulo(x^4 + x + 1)}$$

Since $x = \alpha$ therefore $\alpha^8 = x^2 + 1$, addition and subtraction of two elements (say α^i and α^j) in $GF(q)$ is quite simple because the result will never jump out of the Field and can be achieved by polynomial representation. The summoned result is then transformed as a power of α. But multiplication and division are tedious job, because their end result might jump out of the Field $GF(q)$. For instance in $GF(7)$ if we multiply the elements 3 and 4, it gives end result 12, and hence 12 is not in the Field. To resolve this setback we must perform out arithmetic under $modulo(7)$, so that $(3 \cdot 4) modulo(7) \approx 5$ and hence the end result 5 must recline into the Field. For any two arbitrary elements $i \,\&\, j$, if $i + j < d$, then the Field to be closed under multiplication operation because $\alpha^i \cdot \alpha^j < 2^d$. But if $i + j > d$ then we may write:

$$\boldsymbol{i + j = d + r} \;\; \forall\, \boldsymbol{0 \leq r} < d$$

$$\boldsymbol{\alpha^i \cdot \alpha^j \approx \alpha^{i+j} \approx \alpha^{d+r} \approx \alpha^r}$$

This result ensures that the Field is closed under multiplication operation. On the other hand, for any $i \in \mathbb{Z}\ \forall\, 0 < i < d$, α^{d-i} is the multiplicative inverse of α^i [1]. For instance for $i = 11, \alpha^4$ is the MI of α^{11}. That means$\alpha^4 \cdot \alpha^{11} = 1 modulo\, p(x)$.

VI.PRIMITIVE POLYNOMIAL OVER $GF(\mathbb{P}^m)$

Consider polynomial $r(x) = x^2 + 1$ define over the domain real number $\mathbb{R}$, but its root does not lie in the domain of $\mathbb{R}$. However its root lies on the domain of complex number. Similarly a polynomial $p(x) \in GF(p)[x]$ doesn't have its roots in their Characteristic Field $GF(p)$ however it has its root in the Field $GF(p^n)$, this $GF(p^n)$ is called an "*Extension Field*" of $GF(p)$. Galois Field $GF(2^m)$ is the extension field of prime Field $GF(2)$. Sometimes $GF(2)$ is also referred as binary Field. Binary addition & multiplication are done by bit wise "XOR" and "AND" operation under $modulo(2)$ operation, and they satisfy commutative, associative and distributive law [3, 8]. Since XOR operation return 0 if both the operands are similar and return 1 otherwise, this ensure that the addition & subtraction are same in Galois Field having Characteristic Field $GF(2)$.A polynomial $p(x)$ of degree n over Galois Field $GF(2)$ is symbolized as $p(x) \in GF(2)[x]$ and is defined as.

$$p(x) = a_n x^n, a_{n-1}x^{n-1}, a_{n-2}x^{n-2}, \dots a_{n-r}x^{n-r}, \dots a_2x^2, a_1x, a_0 \;\forall\, 2 < r \leq n$$

Where $a_i \in \{0, 1\}$. For any positive integer m, there are 2^m polynomials are possible each of degree m. For example the polynomials of degree 1 and 2 are shows in Table II.

Table II

Primitive Polynomial of Degree ≤ 2	
$\boldsymbol{n}$	***Possible Polynomials***
1	$x,\ \ x+1$
2	$x^2+x+1,\ \ x^2+x,\ \ x^2+1$

Now let's assume that an element e of $GF(2)$, if $p(e)=0$ then e is said to be a root of polynomial $p(x)$ over $GF(q)$, therefore we can have:

$$x = e$$
$$x = e + e - e$$
$$x + e = 2e$$

Since all the computation are done under $modulo(2)$ operation, so $x+e=0$ this ensure that in $modulo(2)$ operation 1 is equal to -1. That means $x+e$ is a factor of $p(x)$, The criterion for an irreducible polynomial to be a primitive is that "*a polynomial $p(x)$over $GF(2)[x]$ of degree m is irreducible if it has no factor of degree less than m, moreover it is a factor of other polynomial $P(x) = x^d - 1$, where $d = 2^m - 1$* ". For example the polynomial x^3+x+1 is irreducible and have no factor or factor polynomial $f(x)$ of degree less than 3, and also x^3+x+1 is a factor of polynomial $P(x)=x^7-1$ hence it is a *primitive polynomial* [5]. For any degree there must be a primitive polynomial. Primitive polynomials are the minimal polynomial for the primitive elements in the Galois Field. A primitive polynomial $p(x) \in GF(p)[x]$ is always irreducible in $GF(p)[x]$, however an irreducible polynomial need not always be primitive. For example $x^4+x^3+x^2+x+1$ is irreducible but it is not primitive because $x^4+x^3+x^2+x+1|x^5-1$ instead of $x^{15}-1$, which violet the criteria for a polynomial to be primitive. All irreducible polynomial over $GF(2)[x]$ is primitive if it has degree 2, 3 or 5. In $GF(2)[x]$ if the degree of $p(x)$ is greater than 1 and have even number of terms, then it cannot be irreducible, because 1 is it's a root and hence $(x+1)$ is a factor. It is noted that a binary polynomial that is missing alternate terms are not irreducible. Given an irreducible polynomial $p(x) \in GF(p)[x]$of degree m, to test whether it is primitive or not, divide $x^i - 1\ \forall\, m < i < p^m - 1$ from $p(x)$, if no such i exists, it leaves remainder zero, then the polynomial $p(x)$ is said to be a primitive polynomial. Let us assume that α be a root of m degree primitive polynomial $p(x)$ defined over $GF(2)[x]$ then it must be a root of $x^{p^{m-1}}-1$ and $x^{ord(\alpha)}-1$, and have an order p^m-1. The p^m-1 consecutive power of α form a multiplicative group of order p^m-1 [4]. Since α is a nonzero root of $p(x)$ therefore $p(x)|x^m-1$ and this implies that the order of α must be a factor of m, so we may have $ord(\alpha)|m$. All of the roots have same order hence the set of all roots of $p(x)$ makes a conjugacy class with respect to $GF(q)$. Furthermore $p(x)|x^{ord(\alpha)}-1$ if and only if the $ord(\alpha)$ is same as the order of any roots of $p(x)$[6, 11]. For any prime power q and any positive integer n, there exists a primitive polynomial of degree $n$over Galois Field$GF(q)$. There are:

$$\boldsymbol{a_q(n) = \frac{\varphi(q^n-1)}{n}}$$

Primitive polynomials over $GF(q)$, where $\varphi(n)$is the totient function. A polynomial of degree n over the Finite Field $GF(2)$ (i.e., with coefficients either 0 or 1) is primitive if it has polynomial of order 2^n-1. For example, x^2+x+1 has order 3 since.

$$\frac{x+1}{x^2+x+1} = \frac{x+1}{x^2+x+1}(mod2)$$
$$\frac{x^2+1}{x^2+x+1} = 1 + \frac{x+1}{x^2+x+1}(mod2)$$
$$\frac{x^3+1}{x^2+x+1} = x+1(mod2)$$

Putting $q=2$ in equation $a_q(n) = \frac{\varphi(q^n-1)}{n}$ the numbers of primitive polynomials over $GF(2)$ are: $a_2(n) = \frac{\varphi(q^n-1)}{n}$ giving 1, 1, 2, 2, 6, 6, 18, 16, 48 ... for n =1, 2, 3.... The following Table III list of all possible primitive polynomials $modulo(2)$ of orders 1 through 5 is shown in Table III.

Table III

Primitive Polynomial over $GF(2)$ of Degree ≤ 5	
n	***Primitive Polynomials***
1	$1+x$
2	$1+x+x^2$
3	$1+x+x^3, 1+x^2+x^3$
4	$1+x+x^4, 1+x^3+x^4$
5	$1+x^2+x^5, 1+x^3+x^5, 1+x+x^2+x^3+x^5,$ $1+x+x^3+x^4+x^5, 1+x^2+x^3+x^4+x^5,$ $1+x+x^2+x^4+x^5$

Primitive polynomials are widely used in Field element representation, Pseudo-random bit generation and CRC codes. Primitive polynomials over $GF(2)$ is used for pseudorandom bit generation. In fact, every linear feedback shift register with maximum cycle length (which is 2^n-1, where n is the length of the linear feedback shift register) may be built from a primitive polynomial. For example, given the primitive polynomial $x^{10}+x^3+1$, we start with a user specified $10-$bit seed occupying bit positions 1 through 10, starting from the least significant bit. We then take the 10^{th} and 3^{rd} bits, and create a new 0^{th} bit, so that the XOR of the three bits is 0. The seed is then shifted left one position so that the 0^{th} bit moves to position 1 in each clock pulse.

This process can be repeated to generate $2^{10}-1=1023$ pseudo-random bits. In general, for a primitive polynomial of degree n over $GF(2)$, this process will generate a maximum of 2^n-1 pseudo random bits before repeating the same sequence, while non−primitive polynomial produces sequence $< 2^n-1$. One important property is to note that their reciprocal also form primitive polynomial (i.e., they come in pair). Example $1+x^3+x^4$ is degree 4, its reciprocal $1+x+x^4$ i.e., 10011 and 11001, both are primitive. Technically, one can define primitive polynomial using concepts better than Finite Field Theory.

VII. CLOCK

Clock is a signal that is generated when a sequential circuit works. A clock can transmit high signal if and only if the circuit works. The time interval for which a clock is high is same as the time for which it is low. The number of complete cycle emitted in a second is called *frequency* of the clock and is denoted by υ, while the time taken by clock to complete a cycle is called their *time period* and is denoted by T. Figure III shows the working principal of a clock. The ratio of the "*time for which signal reaches at high to the total time*" is called their *duty cycle*. Every clock has their duty cycle exactly equal to 50%.

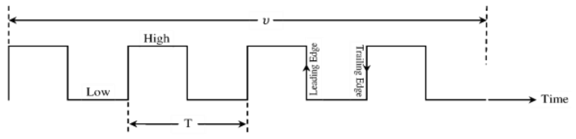


Figure III

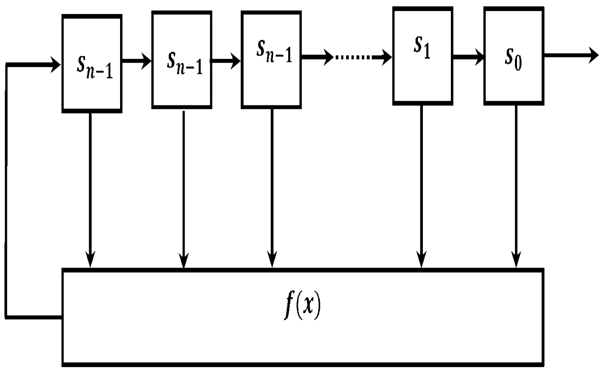
VIII. Q-ARY FSR SEQUENCE

A sequence$s = s_0, s_1$ .... is called a q-ary Feedback Shift Register (FSR) sequence generated by a n stage FSR, with a feedback function $f: \mathbb{F}_{q^n} \to \mathbb{F}_q$ with initial state $s_0, s_1, \dots, s_{n-1}$, if it satisfy the recursion $s_{k+n} = s_k + s_{k-1} + \dots + s_{k+n-1}\ \forall\ k \in \mathbb{Z}^+$. The sequence $\langle s \rangle_{i \in N}$ said to be a q-ary FSR sequence if there exist an $r \in \mathbb{N}$ such that $s_{i+r} = s_r \forall\ i \in \mathbb{Z}^+$, and the sequence is said to be periodic with period r. An FSR sequence with feedback function $f(x)$ is called LFSR, if $f(x)$ is linear, i.e., of the form of

$$f(s_{n-1}, \dots, s_1, s_0) \\ = s_{n-1}k_{n-1}, s_{n-2}k_{n-2}, \dots, s_1k_1, s_0k_0\ \ \forall\ k_i\ \in \mathbb{F}_q$$

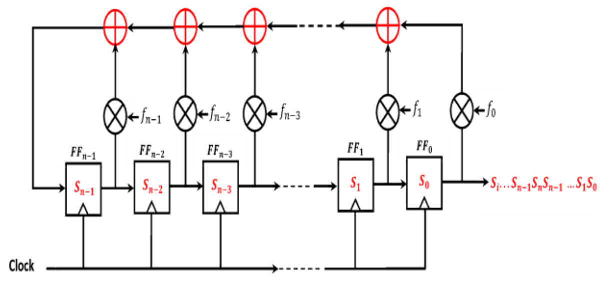
IX. LINEAR FEEDBACK SHIFT REGISTERS

A Mealy machine, Autonomous Linear Feedback Shift Registers (LFSR), Pseudo Random number Generator, Polynomial Sequence Generator, Pseudo Random Pattern Generator or simply an LFSR comprises of two parts: (i) a clock storage elements (Flip-Flop or FF) and (ii) a feedback path. The number of storage elements gives us the degree of the LFSR. In other words, an LFSR with m Flip-Flop is said to be of degree m.The possible m feedback paths compute the input for the left most FF as XOR or XNOR sum of certain Flip-Flop in the shift register. The internal value of LFSR is called initial fill, initial vector or a seed (in mathematical terminology) and because the operation of the register is deterministic, the stream of values produced by the register is completely determined by its current or previous states. The output of the LFSR is one bit at each clock. Likewise the register has a finite number of possible states, it must eventually enter a repeating cycle excluding all zeroes pattern, i.e., once it reaches its final state; it will traverse the sequence exactly as before. However, an LFSR with a well-chosen feedback function can produced a sequence of bits which appears random and has a very long cycle. An LFSR $\langle m|p(x)\rangle$ is singular (i.e., connection polynomial $p(x)$ has degree less than m) then not all output sequence are periodic with period m. However the output sequence is ultimately periodic; that is the sequence obtained by ignoring a certain finite number of terms at the beginning are periodic. The internal state bits are denoted by s_i and are shifted by one unit right with each clock pulse. The rightmost bit gives the current output whereas the leftmost bit is to be computed by a feedback function $f(x)$, which is a XOR sum of some FF values in previous state. Since XOR is a linear operation hence the circuit is called a *linear recurrence*. Whether a feedback path is active or not is determine by a feedback coefficients $f_{m-1}, \dots, f_1, f_0$. If $f_i = 0$ meaning there is no feedback at FF_i and if a feedback exists at location i then we set $f_i = 1$.The value of feedback coefficients f_i play a crucial role for the output sequence produced by the LFSR. Application of LFSR include Test Pattern Generator (TPG), Output Response Analyzers (ORA), Pseudo Random Number (PRN), Error Correcting Code, Pseudo Random Pattern Generation and Signature Analysis in logic Built In Self-Test (BIST), Test Data Decompression and Test Data Compaction in Scan



Compression, Cyclic Redundancy Codes (CRC), shut off the screen if no one touched the keyboard for 2^n-1 seconds, Sends an input to control the computer at every 100ms, reboot if no response, Cycling through the addresses for refreshing a Dynamic Random Access Memory

(DRAM) and many more. However it has some Problems for TPGs [10].



X. TERMINOLOGY RELATED WITH LFSR

Taps, period, internal state, initialization vector IV, lockup state etc. are the essential terminology associated with LFSR that are very important to discuss.

A. Taps

Lines that run from the output of one register within the LFSR into the XOR gates that determine input to the register within the LFSR. Taps are chosen on the basis of primitive polynomials. Only certain combination of taps will produced a maximal length.

B. Period

An m degree LFSR can produce a maximum of $2^m - 1$ distinct sequence of random number, and then it repeats the same sequence. The number of count for which it generates distinct sequence without repetitions is called their period. The LFSR's period depends on the seed value, the tap positions and the feedback type, *OR if r is the smallest positive integer such that $p(x)|x^r + 1$, then integer r is called the period of LFSR* [9].

C. Internal State

At each clock pulse, all the bit are shifted towards MSB from LSB, and then XOR bit is fed into the LSB register of LFSR. Hence it results to change the bit pattern. Each of bit patterns is known as their internal states. The order of states is depend upon your choice to choose the seed value.

D. Seed or Initialization Vector

The initial value of the LFSR is called a seed or Initialization Vector (IV). If the feedback function $f(x)$ involve XOR operation to compute feedback bit, then the registers should be seeded to none zero value.

E. Lockup State

If seed contains all zero initial values, then the LFSR will mire up and will never come in recoverable state, and also won't leave this state. Note that it is possible to design an LFSR that have its lockup state with all ones instead of all zeroes.

XI. WHY LFSR

There are several reasons for which we can prefer Linear Feedback Shift Register (LFSR) than any other registers, some of them are:

- Flip-Flops can be connected by few XOR gates.
- Required less gate consumption.
- Work better than a counter.
- Can be used as a fast counter [1].
- Internal circuit is very fast, Max delay is 1 XOR Delay plus 1 D FF Delay [8].
- Takes less area than any other common counter except a ripple counter [12]
- Much faster than any other common counters except the Mobius counter [10]
- It does not count in binary. It counts $modulo(2^m - 1)$, while binary counter counts $modulo(2^m)$ [16].
- Provides 2^m patterns for n input combinational logic circuit.
- Provides higher clock frequency [2].
- Very little latency and independent of n!.
- Obeys approx. 15 of 25 Standard Statistical Tests [2].

XII. TYPES OF LFSR

Finally for every primitive polynomial there are in fact 4 linear feedback shift register which may be implemented either by using XOR gated in series with each FF output, or with the XOR gate external to the shift register in the feedback path. The external XOR LFSR is called Standard LFSR or Type-I LFSR or External LFSR as shown in the Figure IV. The internal XOR LFSR is called Modular LFSR or Type-II LFSR or Internal LFSR. Each form of LFSR can be made into a signature analyzer by addition of the XOR input to the first D-Type FF [13].

A. Standard LFSR

Following Figure IV shows n stage standard LFSR. It consists of n FF and a number of XOR gates. Since XOR gate are placed on the external feedback path, hence it is also referred as *external XOR LFSR as* shown in Figure IV [14].

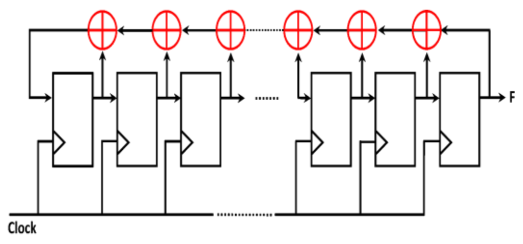


Figure IV. External LFSR

B. Modular LFSR

Similarly, an n stage modular LFSR with each XOR gate placed between two adjacent FF are shown in Figure V, is called an *internal XOR LFSR*, because each stage introduces at most one XOR gate delay. It has higher clock frequency than Standard LFSR as shown in Figure V [14].

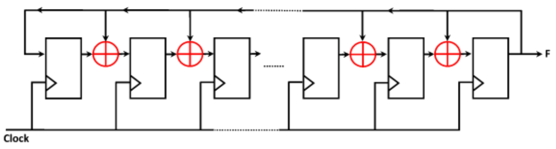
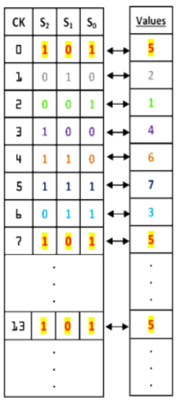


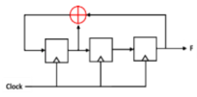
Figure V. Internal LFSR

The sequence generated by the Type-1 and Type-2 LFSR are totally different even if they are seeded with the same initialization vector as shown in the Figure below.

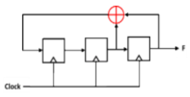


CK	S_2	S_1	S_0		Values
0	1	0	1	⟷	5
1	0	1	0	⟷	2
2	0	0	1	⟷	1
3	1	0	0	⟷	4
4	1	1	0	⟷	6
5	1	1	1	⟷	7
6	0	1	1	⟷	3
7	1	0	1	⟷	5
⋮					⋮
13	1	0	1	⟷	5
⋮					⋮

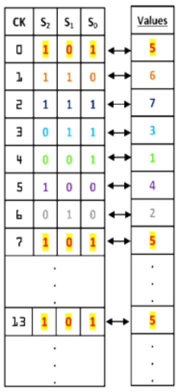
$Table. 5 \langle 1,3 \rangle$



$Fig. 5 \langle 1,3 \rangle$

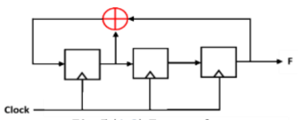


$Fig. 5 \langle 2,3 \rangle$

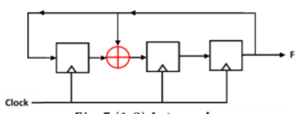


CK	S_2	S_1	S_0		Values
0	1	0	1	⟷	5
1	1	1	0	⟷	6
2	1	1	1	⟷	7
3	0	1	1	⟷	3
4	0	0	1	⟷	1
5	1	0	0	⟷	4
6	0	1	0	⟷	2
7	1	0	1	⟷	5
⋮					⋮
13	1	0	1	⟷	5
⋮					⋮

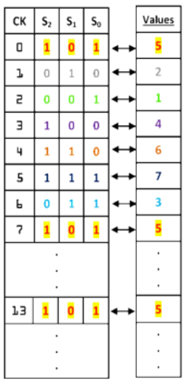
$Table. 5 \langle 2,3 \rangle$



$Fig. 5 \langle 1,3 \rangle \; External$

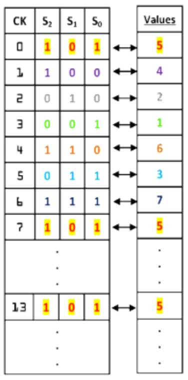


$Fig. 5 \langle 1,3 \rangle \; Internal$



CK	S_2	S_1	S_0		Values
0	1	0	1	⟷	5
1	0	1	0	⟷	2
2	0	0	1	⟷	1
3	1	0	0	⟷	4
4	1	1	0	⟷	6
5	1	1	1	⟷	7
6	0	1	1	⟷	3
7	1	0	1	⟷	5
⋮					⋮
13	1	0	1	⟷	5
⋮					⋮

$Fig. 5 \langle 1,3 \rangle \; External$



CK	S_2	S_1	S_0		Values
0	1	0	1	⟷	5
1	1	0	0	⟷	4
2	0	1	0	⟷	2
3	0	0	1	⟷	1
4	1	1	0	⟷	6
5	0	1	1	⟷	3
6	1	1	1	⟷	7
7	1	0	1	⟷	5
⋮					⋮
13	1	0	1	⟷	5
⋮					⋮

$Fig. 5 \langle 1,3 \rangle Internal$

XIII. DESIGN OF $m - Sequence$ LFSR

An LFSR of size m can result in producing each feasible state throughout the period p which is equal to $2^m - 1$ shift, but it will achieve this period only when appropriate feedback paths have been chosen. For example, an 8 stage LFSR would probably possess a widest possible combination of 1s and 0s after reaching at 255 shifts. Each sequence produced in this shift is a maximal sequence, in general a maximum length sequence. These sequence is usually referred as $m - sequence$ or a Pseudo Noise (PN) or a Pseudorandom Sequences (PS). Maximal length generators can in fact produce two sequences. The first has a length of one, and occurs when the initial state of the generator is set to all zero. The other one has a length of $2^m - 1$. Together, both of these two sequence keep track of all 2^m state of a m-bit state register. Once the feedback taps of an LFSR are non-maximal, the length of the generated sequence relies on the initial state of the LFSR. Each of these sequences is called a State Space of the LFSR.

Mathematical Definition of Galois Field and $m -$ Sequences

To achieve the $m - sequence$ of LFSR, Galois Field are widely used to obtain feedback taps. If a polynomial $p(x)$ of variable x represent as a LFSR, such polynomial is termed as the generator polynomial.

$$G(x) = f_m x^m + f_{m-1} x^{m-1} +, \dots, f_{m-r} x^{m-r}, \dots., f_2 x^2 + f_1 x + f_0$$

$$\forall \, 2 \le r < m$$

The coefficients $f_i \in \{0, 1\}$ signifies the tap weight, 1 for the tap that is connected and 0 otherwise. The order of the polynomial m signifies the number of LFSR stages. Rules of linear algebra apply to the polynomial, however all mathematical operations are performed in $modulo(2)$. For example the generator polynomial $x^3 + x + 1$ represents a LFSR with feedback taps at 3 and 1 denoted as $\langle 1, 3 \rangle$. Now, the second problem is to select feedback taps so that the m-sequences can be produced. The generator polynomial G(x) is said to be primitive if and only if it can't be factored. In order to find such polynomial, it must be a prime number. In this case, when the generator polynomial $G(x)$ is a factor of $x^p + 1$, where $p = 2^m - 1$, it may represent that the LFSR generate a maximal length sequence. Let's take the example of LFSR $\langle 1, 3 \rangle$.

CK	S_4	S_3	S_2	S_1	S_0		Values
0	1	1	1	1	1	↔	1F
1	0	1	1	1	1	↔	0F
2	0	0	1	1	1	↔	07
3	1	0	0	1	1	↔	13
4	1	1	0	0	1	↔	19
5	0	1	1	0	0	↔	0C
6	1	0	1	1	0	↔	16
7	0	1	0	1	1	↔	0B
8	0	0	1	0	1	↔	05
9	1	0	0	1	0	↔	12
10	0	1	0	0	1	↔	09
11	0	0	1	0	0	↔	04
12	0	0	0	1	0	↔	02
13	0	0	0	0	1	↔	01
14	1	0	0	0	0	↔	10
15	0	1	0	0	0	↔	08
16	1	0	1	0	0	↔	14
17	0	1	0	1	0	↔	0A
18	1	0	1	0	1	↔	15
19	1	1	0	1	0	↔	1A
20	1	1	1	0	1	↔	1D
21	0	1	1	1	0	↔	0E
22	1	0	1	1	1	↔	17
23	1	1	0	1	1	↔	1B
24	0	1	1	0	1	↔	0D
25	0	0	1	1	0	↔	06
26	0	0	0	1	1	↔	03
27	1	0	0	0	1	↔	11
28	1	1	0	0	0	↔	18
29	1	1	1	0	0	↔	1C
30	1	1	1	1	0	↔	1E
31	1	1	1	1	1	↔	1F
⋮							⋮
62	1	1	1	1	1	↔	1F
⋮							⋮

Figure VI. PRNG Sequence

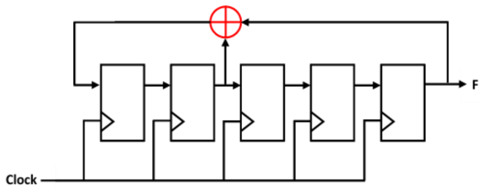


Figure VII. LFSR $\langle 2,5 \rangle$

In this case we will check whether the LFSR produce an $m - sequence$ or not. Initially we keep in mind that $m = 3$ and $p = 2^m - 1$ which is equivalent to 7. It can be shown that its polynomial $x^3 + x + 1$ can never be factored; it is also found that the polynomial is a factor of $x^7 + 1$. Hence, for any primitive polynomial $p(x)$ will really produce a maximal length sequence if and only if;

$$\boldsymbol{p(x)[m] \mid p(x)[2^m - 1]} \text{ and,}$$

$$\boldsymbol{p(x)[m] \nmid p(x)[i] \;\forall\; 1 \le i \le (2^m - 2)}$$

During this demonstration, we went through the procedure for identifying whether the given set of feedback taps would produce a maximal length sequence or not. Normally, we are required to do just the opposite. That is, we are normally required to find all sets of feedback taps that will produce m−sequences for a given register size m. For instance, we may be asked to find all sets of maximal-length feedback taps for an LFSR with $m = 3$. We accomplish this as follows: The length of the $m - sequence$ will be 7. The solution of this problem resides in the primitive factor of polynomial $x^7 + 1$ under $modulo(2)$ operation. The prime factorization of $x^7 + 1$ is something like:

$$\boldsymbol{x^7 + 1 = (x + 1)(\,x^3 + x + 1)(x^3 + x^2 + 1\,)}$$

It should be noted that the size of registers is same as of the order of the primitive polynomials which is equal to m. Out of the three prime factors, the last two meet this criterion. Thus we see that there are exactly two sets of $m{-}sequence$ feedback taps $\langle 1,3 \rangle$ and $\langle 2,3 \rangle$ that exists for degree 3. It should be noted that, there always exist an even number of feedback tap set that produce $m - sequence$ however the size of LFSR may be. For more approval, we can say that for any feedback tap set that produce $m - sequence$ such as $F = \{f_1, f_2, \dots f_i\}$, there must exist a set containing mirror image of the feedback taps of set F which is equivalent to $F' = \{f_1, n - f_{n-1}, \dots, n - f_0\} \;\forall\; 1 \le i \le m$. Here both F and F' are complement to each other under the size of LFSR m. Form this observation we can say that, on subtracting feedback taps number from the size of LFSR i.e., m, one can obtain the set of reverse order taps of all the corresponding elements of F that guarantees to achieve $m - sequence$. Hence we can conclude that, for any set of $m - sequence$ feedback taps, their mirror image feedback taps also produce the mirror image $m - sequence$, and this mirror image sequence is also a $m - sequence$. An astute reader may have noticed that, the $m - sequence$ feedback taps $\langle 1,3 \rangle$ and $\langle 2,3 \rangle$ are mirror image to one another, and hence they both produced the same $m - sequence$. PN sequence have advantageous feature from the computational viewpoint [15]. Due to only these structural properties, PN sequence have enormous applications like Direct Sequence Spread Spectrum (DSSS), Built-In-Self Test (BIST), Decryption Encryption System (DES) and many more [12, 16].

Figure VI shows pattern produced by the LFSR $\langle 2,5 \rangle$ shown in Figure VII. Assume that the LFSR is initially seeded with vector 11111, we observing that each PN sequence s_i has same period $p = 31$, and hold the same properties as illustrated below. To understand properties of PN sequence, please refer to Figure VIII.

Property I: In every period $p = 2^m - 1$, the sequence contains exactly 2^{m-1} number of ***ones***.

Property II: In every period $p = 2^m - 1$, the sequence contain exactly $2^{m-1} - 1$ number of ***zeroes***. This means the total number of ***one's*** is equal to the total number of ***zeroes*** +1, this is called there

Balance Property.

Property III: In every period $p = 2^m - 1$, the sequence has an occurrence of ***one's*** exactly m times in succession.

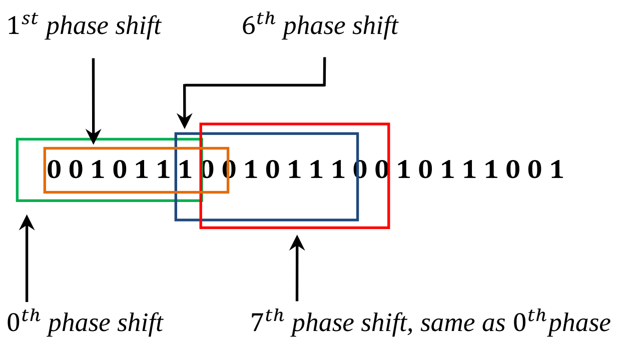
Property IV: In every period $p = 2^m - 1$, the sequence does not have any occurrence of ***zeroes*** exactly m times in the succession.

Property V: In every period $p = 2^m - 1$, the sequence does not have any occurrence of ***one's*** exactly $m - 1$ times in the succession.

Property VI: In every period $p = 2^m - 1$, the sequence has an occurrence of total number of ***zeroes*** exactly $m - 1$ times in succession.

Property VII: Sum of two cycle shifted upto $m - sequences$ is another cycle shift, of the same $m - sequence$. This property is also called Shift and adds property.

Property VIII: If a window of width w slides along a PN sequence for $2^m - 1$ shift, each w tuple except all zeroes w tuple will appear exactly once as shown below.



Four windows of same size w slides along the PN sequences; at each phase shift it shows a unique sequence of bits, each sequence is a PN sequence.

Property IX: As we can observe that the sequence has a Jigsaw Fit of pattern. The upper triangular bit pattern of ones from clock 0 to 4 have a clear Jigsaw Fit with the lower triangular bits pattern of ones from clock 27 to 30 (refer to Figure VI)

Property X: We define a term RUN of $\boldsymbol{r}$ consecutive identical bits. It is a succession of items of the same class enclosed within distinct bits. In a PN sequence we have; Figure VIII shows the different runs of LFSR $\langle 4|3,4\rangle$.

Following observation has been made:

- A run of one's of length $\boldsymbol{r}$.
- A run of zeroes of length $\boldsymbol{r - 1}$.
- A run of one's & a run of zeroes of length $\boldsymbol{r - 2}$.
- Two run of **1**'s & two run of **0**s of length $\boldsymbol{r - 3}$.
- Four run of **1**'s & four run of zeroes of length $\boldsymbol{r - 4}$.

....

- $\boldsymbol{2^{r-3}}$ run of one's and $\boldsymbol{2^{r-3}}$ run of **0**s of length 1 [2].

$modulo(2^m - 1)$

PS	PN
0^{th}	0010111
1^{th}	0101110
2^{th}	1011100
3^{th}	0111001
4^{th}	1110010
5^{th}	1100101
6^{th}	1001011
7^{th}	0010111

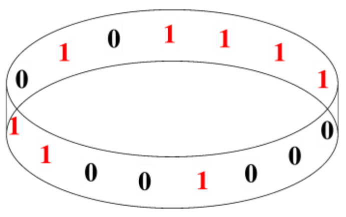


Figure VIII

PS: Phase Shift, PN: Pseudo Number

Synthesis: From the above ideas, we can synthesize that a PN sequence of length $2^m - 1$ contain 2^{i-1} run of $m - i - 1$ one's as well as zeroes $\forall\, 1 \le i \le m - 2$ as provide in Table IV below.

Table IV.

#Run	Successi on	Runs	Synthesi s	Property
1	4	1111	No	3
N	4	0000	No	4
N	3	111	No	5
1	3	000	No	6
1	2	11	Yes	No
1	2	00	Yes	No
2	1	1	Yes	10.6
2	1	0	Yes	10.6
Number of $\boldsymbol{1's = 8}$, and $\boldsymbol{0's = 7\ m = 4}$				

On the basis of above properties, some conclusions are drawn for an ideal PN sequence as given below. Every PN sequence must hold;

A. Balance Property

A PN sequence must have equal number of one's and zeroes, and should have no DC component to avoid a spectral spike at DC or biasing the noise in dispreading.

B. Run Length Property

The run lengths are generally short, and it is observed that half of the runs are of length 1. A fraction $\frac{1}{2^n}$ of all runs are of length n. Long runs reduce the BW spreading and it's an advantage over PN sequence.

C. Shift Property

If the sequence are shifted by a non-zero number of elements/bits, the resulting sequence will have half of its sequence exactly same as of the original sequence, while the half of it is totally different from the original sequence.

Note: A deterministic sequence that holds ***Balance, Run Length and Shift*** grows ***Asymptotically Large,*** this property is referred to as a ***Pseudo Noise*** or ***Noiselike*** signals. $m-sequence$, $Quaternary\ sequence$, $Gold\ Codes$, $Walsh\ functions$ and $Kasami\ sequence$ are the examples of PN sequences.

XIV. CONCLUSION

The paper presented a brief analysis and design of LFSR based $m-sequence$ PRNG that are very useful to implement cryptographic primitives. Some important mathematical tools such as Field, Galois Field, Primitive Polynomial and Primitive Polynomial over Galois Field and LFSR have been discussed. The paper provide a design of $m-sequence$ PRNG that follows the security measures as described by NIST Standard. In future work, it is proposed to implement LFSR based cryptographic primitives that are used in Key Exchange and Data Encryption by using the said mathematical tools.

REFERENCES

1. National Institute of Standards and Technology, Advanced Encryption Standard, FIPS 197 (2011).
2. Fernandes, Rebecca Angela, and Niju Rajan. "Power Optimization of Linear Feedback Shift Register (LFSR) using Power Gating." Power 5.05 (2018).
3. D. A. Cox, Galois Theory, 2nd ed., Wiley, Hoboken, 2012.
4. D. A. Cox, Evariste Galois and Solvable Permutation Groups,http://www.cs.amherst.edu/∼dac/lectures/bilbao.pdf.
5. G. Frei, The Unpublished Section Eight: On the Way to Function Fields over a Finite Field, pp. 159–198 in "The Shaping of Arithmetic after C. F. Gauss's Disquisitiones Arithmeticae," ed. C. Goldstein, N. Schappacher, J. Schwermer, Springer-Verlag, Berlin, 2007.
6. E. H. Moore, A Doubly-Infinite System of Simple Groups, pp. 208–242 in "Mathematical papers read at the International Mathematical Congress held in connection with the World's Columbian Exposition, Chicago, 1893," Macmillan & Co., New York, 1896.
7. Mashhadi, Samaneh, and Massoud Hadian Dehkordi. "Two verifiable multi secret sharing schemes based on nonhomogeneous linear recursion and LFSR public-key cryptosystem." Information Sciences 294 (2015): 31-40.
8. Tan, Zuxiong, et al. "A New Pseudo-Random Number Generator Based On The Leap-Ahead LFSR Architecture." 2018 IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA). IEEE, 2018.
9. Mo, Hongjia, and Michael Peter Kennedy. "Influence of Initial Conditions on the Fundamental Periods of LFSR-Dithered MASH Digital Delta–Sigma Modulators with Constant Inputs." IEEE Transactions on Circuits and Systems II: Express Briefs64.4 (2016): 372-376.
10. Tzanakis, Georgios, et al. "Constructing new covering arrays from LFSR sequences over Finite Fields." Discrete Mathematics 339.3 (2016): 1158-1171.
11. Panda, Amit Kumar, Praveena Rajput, and Bhawna Shukla. "FPGA implementation of 8, 16 and 32 bit LFSR with maximum length feedback polynomial using VHDL." 2012 International Conference on Communication Systems and Network Technologies. IEEE, 2012.
12. Ahmad, Afaq, Sayyid Samir Al-Busaidi, and Mufeed Juma Al-Musharafi. "On Properties of PN Sequences generated by LFSR a Generalized Study and Simulation Modeling." Indian Journal of Science and Technology 6.10 (2013): 5351-8.
13. Tsoi, Kuen Hung, K. H. Leung, and Philip Heng Wai Leong. "Compact FPGA-based true and pseudo random number generators." 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2003. FCCM 2003.. IEEE, 2003.
14. Arnault, François, and Thierry P. Berger. "Design and properties of a new pseudorandom generator based on a filtered FCSR automaton." IEEE Transactions on Computers 54.11 (2005): 1374-1383.
15. Tsalides, Ph, T. A. York, and A. Thanailakis. "Pseudorandom number generators for VLSI systems based on linear cellular automata." IEE Proceedings E (Computers and Digital Techniques) 138.4 (1991): 241-249.
16. Deepthi, P. P., and P. S. Sathidevi. "Design, implementation and analysis of hardware efficient stream ciphers using LFSR based hash functions." Computers & Security 28.3-4 (2009): 229-241.

AUTHORS PROFILE

Shabbir Hassan is a Research Scholar at the Aligarh Muslim University, Aligarh. He holds Master in Computer Science and Applications (MCA) and currently pursuing Ph.D at Department of Computer Science, Aligarh Muslim University. His thrust area is "Analysis and Design of Lightweight Stream Cipher" and area of interest includes Applied Mathematics, Analysis and Design of Algorithms, Dynamic Programming, Network Security and Cryptography. He has qualified UGC-National Eligibility Test (NET) and has availed Junior Research Fellowship (JRF) during the Research Work. Throughout his career, he has been involved in innovative Software Development and Academic Teaching of Computer Science subjects like C, JAVA, Python, Data Structure, Operating System, Automata Theory and Computer Networks. He has presented his research work in several National and International IEEE Conferences and marked his active participation in many Conferences, Workshops and Symposia. His research papers have published in many reputed peer reviewed Journals of International repute like Springer, Elsevier, JASRAE, InderScience and Scopus Indexed Database. Apart from the Academic Research and Software Development, he is enriched with the passion of poetry and philosophy and engages himself in Social Works.

Prof. Mohammad Ubaidullah Bokhari is presently working as Professor in the Department of Computer Science, AMU, Aligarh, (INDIA) and Principal Investigator (PI) of the ambitious project, NMEICT ERP Mission Project (Govt. of India). He has also worked as an Associate Professor and Director of Studies in Australian Institute of Engineering & Technology, Victoria, Melbourne (Australia). Prof. Bokhari has a vast teaching experience of more than twenty three years. Under his guidance more than 700 students of PG level has completed projects/Dissertations as well as more than 200 projects at UG level.
Prof. Bokhari obtained his doctorate in the field of Software Reliability and Master's degree in computer science from Aligarh Muslim University. During his academic pursuit he has visited different countries like Australia, France, Canada, South Africa, etc. for research purpose and acquiring academic excellence. He is the recipient of many scholarships throughout his career. He is recipient of Merit Scholarship from high school to graduation level. He was awarded prestigious Australian Postgraduate Award (Industry) Scheme Scholarship in 2004. He is lifetime member of Computer Society of India (C.S.I.) and member of IEEE. He has published more than 100 research papers in reputed National and International Conferences.