Theoretical and Applied Informatics
Vol. 29 (2017), no. 1-2, pp. 1–13
DOI: 10.20904/291-2001

Xorshift random number generators from primitive polynomials

Susil Kumar Bishoi[1,2*] Surya Narayan Maharana[3†]

[1]Center for Artificial Intelligence and Robotics,
Defence Research and Development Organisation,
CV Raman Nagar, Bengaluru 560093, India

[2]Faculty of Informatics,
Masaryk Univerzity, Czechia

[3]Indian Institute of Technology,
Ropar, Punjab 140001, India

Abstract A class of Xorshift Random Number Generators (RNGs) are introduced by Marsaglia. We have proposed an algorithm which constructs a primitive Xorshift RNG from a given primitive polynomial. We also have shown a weakness present in those RNGs and suggested its solution. A separate algorithm also proposed which returns a full periodic Xorshift generator with desired number of Xorshift operations.

Keywords random number generators; Xorshift generator; primitive polynomials; linear feedback shift registers; multiple-recursive matrix method;

Received 08 AUG 2017 **Revised** 31 DEC 2017 **Accepted** 31 DEC 2017

1 Introduction

Random bits are required in many areas including in cryptography, computer simulation, statistical sampling, etc. A True Random Number Generator (TRNG) can be used to generate these random bits. However, the TRNG design uses some uncontrollable physical processes as a source of true randomness and in most practical environments this is an inefficient procedure. So, a Pseudo Random Number Generator (PRNG) can be used in place of a TRNG. PRNG takes a small bit length seed (random) as input and produces a very large binary sequence which appears to be random. The concept of PRNG motivates the design of stream ciphers and in stream cipher

*E-mail: skbishoi@cair.drdo.in
†E-mail: suryan.math@gmail.com

design, Linear Feedback Shift Register (LFSR, see Golomb [1], Lidl and Niederreiter [2]) is used as one of the important basic building blocks.

The LFSR is very popular in hardware as it has fast and low cost of implementation. If it is primitive, then it produces maximum length periodic bitstream for any nonzero initial state. Also, bitstream generated by the LFSR have very good statistical properties. However, it produces only one new bit in each cycle, so such ciphers are often referred as bit-oriented ciphers and could not take the advantage of available word based modern operations. However, the word based LFSR called MRMM [3, 4, 5, 6] takes this advantage. By Zeng at el., it is called as σ-LFSR [7]. It is shown in [8, 9] that Marsaglia's Xorshift RNGs are special case of the MRMMs.

In this paper, we have given an algorithm which constructs Xorshift RNGs from a binary primitive polynomial. Later, we have found a weakness in these RNGs generated from this algorithm and suggested a solution to overcome this weakness. The paper is organized as follows. In Sec. 2, we introduce some notations, definitions and results concerning to the primitive LFSRs and Xorshift generators. We propose the construction algorithm for Xorshift RNGs in Sec. 3. In Sec. 4, some results and issues pertaining to the Xorshift RNGs produced by construction algorithm are discussed. Finally, conclusion is made in Sec. 5.

2 Notation and theory

Let $\mathbb{F}_q$ denote the finite field with q elements, where q is a prime power and $\mathbb{F}_q[\mathrm{X}]$ be the ring of polynomials in one variable X with coefficients in $\mathbb{F}_q$. Denote $M_m(\mathbb{F}_q)$ the set of all $m \times m$ matrices with entries in $\mathbb{F}_q$ and $\mathrm{GL}_m(\mathbb{F}_q)$ be the set of all $m \times m$ invertible matrices. For $C \in M_m(\mathbb{F}_q)$, C_{ij} denotes the entry of the matrix C at i^{th} row and j^{th} column. For any square matrix C, $\det(C) = |C|$ denotes its determinant whereas C^T denotes the transpose of the matrix C. The notation $\mathrm{ord}(C)$ denotes the period of the matrix C. $\lceil n \rceil$ denotes least positive integer greater than or equal to n. Let $R \in M_m(\mathbb{F}_2)$ be the right shift operator defined as $Rx = (0, x_1, x_2, \ldots, x_{m-1})^T$, where $x = (x_1, x_2, \ldots, x_m)^T \in \mathbb{F}_2^m$. Then the matrix form of R is as follows

$$R = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}_{m \times m} \tag{1}$$

Similarly, let L be the left-shift operator defined as the transpose of the matrix R, i.e., $Lx = L(x_1, x_2, \ldots, x_m)^T = (x_2, x_3, \ldots, x_m, 0)^T$. For a positive integer k, L^k means L is applied for k times i.e., k is the amount of shifting in left direction and R^k is defined similarly. It is easy to see that both $L^k x$ and $R^k x = 0$ if $k \geq m$. Let $I_m \in \mathrm{GL}_m(\mathbb{F}_q)$ be the identity matrix.

2.1 LFSR

A sequence $s_0, s_1, s_2, \ldots$ with elements from a finite field $\mathbb{F}_q$ is called periodic if there exists a nonnegative integer p such that $s_{i+p} = s_i$ for all $i \geq 0$. The smallest such integer p is called the

period of the sequence. For a periodic sequence, it is always possible to have a relation called linear recurring relation (LRR) [2] among the elements as

$$s_{i+n} = -(c_0 s_i + c_1 s_{i+1} + \cdots + c_{n-1} s_{i+n-1}) \tag{2}$$

where $c_i \in \mathbb{F}_q$ and the integer n is called the degree of the LRR. It is well known that for a given periodic sequence in $\mathbb{F}_q$ there is a minimum degree LRR which satisfy the periodic sequence. The associated polynomial $f(x) = x^n - c_{n-1}x^{n-1} - \ldots - c_1 x - c_0$ is called the characteristic polynomial of the LRR. The companion matrix T of the polynomial f is as follows

$$T = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ c_0 & c_1 & c_2 & \ldots & c_{n-1} \end{pmatrix}_{n \times n} \tag{3}$$

If the column vector $S_0 = (s_0, s_1, \ldots, s_{n-1})^T \in \mathbb{F}_q^n$ is the initial states of the LFSR, then $S_1 = T(S_0) = (s_1, s_2, \ldots, s_n)^T$ where s_n is calculated using the equation (2). The successive states of the LFSR are obtained by repeated application of T. If S_k be the states of the LFSR after k^{th} iteration, then $S_k = T^k(S_0)$. Again, it is proved that the sequence generated by the LRR have period $(q^n - 1)$ if and only if the polynomial associated with the LRR is a primitive polynomial of degree n over the field $\mathbb{F}_q$ [2, 10] .

The primitive LFSRs have very nice properties. The primitive LFSRs produce bit sequence which not only have a large period, but also have good statistical properties required for cryptographic applications. Again, they have low cost of implementation in hardware [10, 11]. So, the LFSRs are quite useful in generation of pseudorandom bit sequences. However, LFSR produces only one new bit per cycle and in many situations such as high speed link encryption, an efficient software encryption technique is required. In such cases, bit-oriented ciphers do not provide adequate efficiency. In case of the LFSR of order n, total n shifting along with the feedback computation is needed to produce one bit output. Thus, an LFSR takes $O(n)$ bit manipulations in order to produce only a single bit. Therefore, in case of software implementation point of view, the LFSR does not take the advantage of the available word based modern processors. However, the word based RNGs like σ-LFSRs [12, 13] and Xorshift RNGs [14] take this advantage.

2.2 Xorshift generator

Xorshift generator [14] introduced by Marsaglia is a linear operator T, which uses only two word based operations called shifting (both right and left) and exclusive-or (XOR). The basic idea of Xorshift generators is that the state is modified by applying repeatedly shift and XOR operations. If $\mathbf{S}_0 = (\mathbf{s_0}, \mathbf{s_1}, \ldots, \mathbf{s_{n-1}})^T \in \mathbb{F}_2^{mn}$ is the initial seed, where each $\mathbf{s_i}$ is m-bit in size, then $\{T\mathbf{S}_0, T^2\mathbf{S}_0, T^3\mathbf{S}_0, \ldots\}$ is the sequence of words generated by T. Note that, in case of Xorshift RNGs, $T\mathbf{S}$ can be computed using a small number of Xorshift operations for any $\mathbf{S} \in \mathbb{F}_2^{mn}$. Here,

the companion matrix of this operator T in the block form is

$$T = \begin{pmatrix} \mathbf{0} & I_m & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I_m & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & I_m \\ (1+L^a)(1+R^b) & \mathbf{0} & \mathbf{0} & \cdots & (1+R^c) \end{pmatrix} \tag{4}$$

where a, b and c are three positive integers and each block is an $m \times m$ matrix. Here $\mathbf{0}$ is the $m \times m$ zero matrix and $T\mathbf{S}_0 = (\mathbf{s_1}, \mathbf{s_2}, \ldots, \mathbf{s_{n-1}}, A\mathbf{s_0} + B\mathbf{s_{n-1}})^T$, where $A = (1 + L^a)(1 + R^b)$ and $B = (1 + R^c)$. So its implementation requires only a few number of Xorshift operations per pseudo-random number generation. Again, the Xorshift generators have implementation advantages when the size of the each state in bits is a multiple of the computer word size m (typically $m = 32$ or 64). The Xorshift RNGs are extremely fast and there are several values of triplet (a, b, c) for which the companion matrix T has maximal period. Marsaglia [14] lists all those triplets (a, b, c) that yield maximal period Xorshift generators with $m = 32$ and $m = 64$. Later it was verified by Panneton and L'Ecuyer [9] and also shown some deficiencies after analyzing this class of generators. Brent also discussed a potential problem related to correlation of outputs with low Hamming weights and suggested a technique to overcome that problem [8].

From Eq. (4), it is clear that the dimension of the matrix T is $mn \times mn$ and so the maximal period of T could be $2^{mn} - 1$. Let $P(z) = \det(T - Iz)$ be the characteristic polynomial of T, then T is full periodic (i.e., $\mathrm{ord}(T) = 2^{mn} - 1$) if and only if $P(z)$ is a primitive polynomial over the binary field $\mathbb{F}_2$[2, 10]. The list of triplet (a, b, c) were listed out as follows:

- It first constructs the matrix T using the triplet(a, b, c) as in Eq. (4).
- Checks the primitiveness of the characteristic polynomial $P(z)$ of the matrix T.
- If $P(z)$ is primitive, then the triplet (a, b, c) is added to the list.

In this process, to get one such triplet it needs several attempts for primitiveness checking of the polynomial $P(z)$. Now, we are proposing an algorithm which does the reverse i.e., it first finds a primitive polynomial and then constructs a Xorshift generator T from this primitive polynomial. The construction algorithm is described in the following section.

3 Construction algorithm for Xorshift RNG

In this section, we present the algorithm which constructs a Xorshift RNG from a given primitive polynomial. Let $f(X) = \sum_{i=0}^{mn} a_i X^i$ be a polynomial of degree mn over $\mathbb{F}_2$. Then using the coefficients a_i's define n number of $m \times m$ matrices C_i for $i = 1, 2, \ldots, n-1$, where first $(m-1)$

columns contain zeros only and is as below

$$C_i = \begin{pmatrix} 0 & 0 & \dots & 0 & a_i \\ 0 & 0 & \dots & 0 & a_{n+i} \\ 0 & 0 & \dots & 0 & a_{2n+i} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & a_{(m-1)n+i} \end{pmatrix}_{m \times m} \tag{5}$$

Again, define the matrix C_0 in the following form

$$C_0 = \begin{pmatrix} 0 & 0 & \dots & 0 & a_0 \\ 1 & 0 & \dots & 0 & a_n \\ 0 & 1 & \dots & 0 & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & a_{(m-1)n} \end{pmatrix}_{m \times m} \tag{6}$$

Now, using the matrix coefficients $C_0, C_1, \dots, C_{n-1}$, constructs the matrix T of size $mn \times mn$ as given in the Eq. (7)

$$T = \begin{pmatrix} \mathbf{0} & I_m & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I_m & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & I_m \\ C_0 & C_1 & C_2 & \cdots & C_{n-1} \end{pmatrix}. \tag{7}$$

Then, for the column vector $\mathbf{S} = (\mathbf{s_0}, \mathbf{s_1}, \dots, \mathbf{s_{n-1}})^T$

$$T\mathbf{S} \;=\; (\mathbf{s_1}, \mathbf{s_2}, \dots, \mathbf{s_{n-1}}, C_0\mathbf{s_0} + C_1\mathbf{s_1} + \dots + C_{n-1}\mathbf{s_{n-1}})^T \tag{8}$$

Let $M(X) = I_m X^n - C_{n-1}X^{n-1} - \cdots - C_1 X - C_0$. Then $M(X)$ is an $m \times m$ matrix polynomial. We call $M(X)$ as the matrix polynomial corresponding to the polynomial $f(X)$. Using the following results, it is possible to calculate the determinant of an $mn \times mn$ matrix from the determinant of an $m \times m$ matrix [12, lemma 2.3].

Lemma 1. *Let T be the matrix corresponding to the polynomial $f(X)$ of degree mn as defined in Eq. (7). Then the characteristic polynomial of T is equal to the determinant of $M(X)$.*

Lemma 2. *Let $M(X)$ be the matrix polynomial corresponding to the polynomial $f(X)$ of degree mn over $\mathbb{F}_q$. Then the determinant $|M(X)|$ is equal to $f(X)$.*

Proof. The matrix form of $M(X)$ is

$$M(X) = \begin{pmatrix} X^n & 0 & 0 & \cdots & 0 & f_0 \\ -1 & X^n & 0 & \cdots & 0 & f_1 \\ 0 & -1 & X^n & \cdots & 0 & f_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & X^n & f_{m-2} \\ 0 & 0 & 0 & \cdots & -1 & f_{m-1} + f_m X^n \end{pmatrix}_{m \times m} \tag{9}$$

where $f_i(X) = \sum_{k=in}^{(i+1)n-1} a_k X^{k-in}$ for $i = 0, 1, \ldots, (m-1)$ and $f_m(X) = a_{mn} \neq 0$. Multiply X^n with the n^{th} row and add to the $(n-1)^{\text{th}}$ row of the above matrix $M(X)$. This will remove X^n from the $(n-1)^{\text{th}}$ row without any change in the determinant. Now, add X^n times the new $(n-1)^{\text{th}}$ row to the $(n-2)^{\text{th}}$ row. This will remove X^n from the $(n-2)^{\text{th}}$ row. Continue this procedure till all the X^n terms on the main diagonal have been removed. Then, the resultant matrix will have the same determinant as $M(X)$ and it will be in the following form

$$\begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & g_0 \\ -1 & 0 & 0 & \cdots & 0 & g_1 \\ 0 & -1 & 0 & \cdots & 0 & g_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & g_{m-2} \\ 0 & 0 & 0 & \cdots & -1 & g_{m-1} \end{pmatrix}_{m \times m} \tag{10}$$

where

$$\begin{aligned} g_0 &= f_0 + X^n \left(f_1 + X^n \left(f_2 + \cdots + X^n \left(f_{m-1} + X^n f_m\right) \cdots\right)\right) \\ g_1 &= f_1 + X^n \left(f_2 + \cdots + X^n \left(f_{m-1} + X^n f_m\right) \cdots\right) \\ g_2 &= f_2 + \cdots + X^n \left(f_{m-1} + X^n f_m\right) \\ &\vdots \\ g_{m-2} &= f_{m-2} + X^n \left(f_{m-1} + X^n f_m\right) \\ g_{m-1} &= f_{m-1} + X^n f_m \end{aligned} \tag{11}$$

After suitable operations, it can be shown that

$$\det\left(M(X)\right)) = \det \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & g_0 \\ -1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 0 \end{pmatrix} = (-1)^{2(m-1)} g_0. \tag{12}$$

But $g_0 = f(X)$ and thus proves the lemma. □

From Lemma 1 and Lemma 2, it is clear that the characteristic polynomial of T is primitive if the polynomial $f(X)$ is primitive. Therefore, if $f(X)$ is primitive, then T is full periodic i.e., $\text{ord}(T) = (2^{mn} - 1)$ [2, 10]. Our next goal is to show that the matrix operator T belongs to the class of Xorshift RNGs. Note that all the matrix coefficients C_i given in Eq. (5) and (6) used in Eq. (8) have a special form. The construction algorithm for Xorshift RNGs takes the advantage of these special structures. It is easy to see that the matrix C_0 can be written as $C_0 = R + \widehat{C_0}$, where R is the right shift operator and $\widehat{C_0}$ is an $m \times m$ matrix having first $(m-1)$ zero columns. Again, the last column of both $\widehat{C_0}$ and C_0 are same and the structure of $\widehat{C_0}$ is exactly same as C_j, for $j \geq 1$. Because of the following lemma, we will show that $T\mathbf{S}$ can be computed using only Xorshift operations.

Input: A primitive polynomial $f(X)$ of degree mn over $\mathbb{F}_2$.
Output: A Xorshift RNG of order n over $\mathbb{F}_{2^m}$.
1: Construct the matrix coefficients C_is as in Eq. (5) and (6)
2: Construct the matrix T as described in the Eq. (7)
3: Return the matrix T

Algorithm 1 Construction of primitive Xorshift RNGs

Lemma 3. *[12] For any matrix $A \in M_m(\mathbb{F}_2)$ having all the columns zero except the m^{th} column and for any vector $\mathbf{s} = [s_0, s_1, \ldots, s_{m-1}]^T \in \mathbb{F}_2^m$, we have*

$$A\mathbf{s} = s_{m-1}\mathbf{v}_m \tag{13}$$

where $\mathbf{v}_m$ represents the m^{th} column of the matrix A.

By invoking Lemma 3, $T\mathbf{S}$ can be rewritten as follows:

$$T\mathbf{S} = (\mathbf{s_1}, \mathbf{s_2}, \ldots, \mathbf{s_{n-1}}, \mathbf{s_n})^T \tag{14}$$

where,

$$\mathbf{s_n} = (R\mathbf{s}_i + \alpha_0\mathbf{v}_0 + \alpha_1\mathbf{v}_1 + \cdots + \alpha_{n-1}\mathbf{v}_{n-1}) \tag{15}$$

and α_i is the least significant bit (LSB) of $\mathbf{s}_i$ and $\mathbf{v}_i$ is the m^{th} column of the matrix C_i ($0 \leq i \leq n-1$), that is $\mathbf{v}_i = [a_i, a_{n+i}, \ldots, a_{(m-1)n+i}]^T$. It is clear that the Eq. (15) can be computed by using only one right shift operation and at most n XOR operations and thus, it falls into the class of Marsaglia's Xorshift RNGs. We call Eq. (15) as feedback computation function.

Now we are in a position to propose the construction algorithm for Xorshift RNGs. The sequential steps of the construction algorithm are described in Alg. 1.

The complexity of the Alg. 1 is $O(1)$ as it generates a primitive Xorshift generator T from a given primitive polynomial just by expressing the coefficients in matrix form.

Lemma 4. *The primitive Xorshift RNGs of order n over $\mathbb{F}_{2^m}$ generated by Alg. 1 will have at least two and at most $(n+1)$ Xorshift operations in the feedback function computation.*

Proof. Alg. 1 generates primitive Xorshift RNGs from the primitive polynomial and again, the constant term of the primitive polynomial must be nonzero i.e., $a_0 \neq 0$. This implies $\mathbf{v}_0 \neq 0$ as $\mathbf{v}_0 = [a_0, a_n, \ldots, a_{(m-1)n}]^T$. Again, in the recurrence relation in Eq. (15), the right shift R will be present irrespective of any polynomial. So there will be at least two Xorshift operations in the feedback computation. Also, in Eq. (15), there are almost $(n+1)$ nonzero terms. This completes the proof. □

4 A note on construction algorithm for Xorshift generators

The Xorshift generators generated from the primitive polynomials and so are full periodic. For every primitive polynomial of degree mn, it constructs distinct Xorshift generator. Therefore,

Iteration No.	States of Xorshift generator
1	$(d, \overbrace{\mathbf{0}, \mathbf{0}, \cdots, \mathbf{0}}^{\text{(n-1) times}})$
2	$(\mathbf{0}, d, \overbrace{\mathbf{0}, \mathbf{0}, \cdots, \mathbf{0}}^{\text{(n-2) times}})$
3	$(\mathbf{0}, \mathbf{0}, d, \overbrace{\mathbf{0}, \mathbf{0}, \cdots, \mathbf{0}}^{\text{(n-3) times}})$
$\vdots$	$\vdots$
n	$(\overbrace{\mathbf{0}, \mathbf{0}, \cdots, \mathbf{0}}^{\text{(n-1) times}}, d)$
$n+1$	$(\frac{d}{2}, \overbrace{\mathbf{0}, \mathbf{0}, \cdots, \mathbf{0}}^{\text{(n-1) times}})$
$n+2$	$(\mathbf{0}, \frac{d}{2}, \overbrace{\mathbf{0}, \mathbf{0}, \cdots, \mathbf{0}}^{\text{(n-2)times}})$
$\vdots$	$\vdots$

Table 1 States of Xorshift RNG

total number of full periodic Xorshift generators of order n over the field $\mathbb{F}_{2^m}$ produced by this algorithm is $\frac{\phi(2^{mn}-1)}{mn}$.

4.1 Weakness in Initialization of Xorshift generator States

The primitive Xorshift RNGs generated by the construction algorithm have efficient software implementation property, however from cryptographic point of view they have a weakness similar to the Lagged Fibonacci Generator (LFG) [8, 15]. In LFG, if all states are initialized with even numbers, then the feedback value will be always even in every iteration. To counter this weakness, at least one state of the LFG must be initialized with an odd value. Similar kind of weakness is also present in the Xorshift RNGs constructed by Alg. 1. If first $(n-1)$ states (i.e., $\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{n-2}$) of the Xorshift RNG generated by the construction algorithm are even, then there will be only one active term in the feedback value computation i.e., $R\mathbf{s}_i$. This happens because the α_i defined in Eq. (15) is the least significant bit (LSB) of the state $\mathbf{s}_i$ and so equal to zero for even value of $\mathbf{s}_i$. If all states are multiple of 2^l i.e., $\mathbf{s}_i = 2^l k_i$, then there will be only one active component in the feedback function computation till nl many iterations and for $0 < j < nl$, $\mathbf{s}_{n+j} = R^{j_1}\mathbf{s}_{j_2}$, where $j_1 = \lceil \frac{j}{n} \rceil$ and $j_2 = (j-1) \bmod n$.

In particular, if the states $\mathbf{s}_i$ for $0 \leq i < (n-1)$ are zero vectors and $\mathbf{s}_{n-1} = d$, where d is a multiple of 2^l, for some integer $l > 0$, then the Tab. 1 gives the states of Xorshift generator after the subsequent iterations.

If the content of stage 0 is the output word in each iteration, then the first nl words of the

output sequence produces by the Xorshift generator is as follows

$$\overbrace{\mathbf{0},\mathbf{0},\cdots,\mathbf{0}}^{\text{(n-1) times}},\frac{d}{2},\overbrace{\mathbf{0},\mathbf{0},\cdots,\mathbf{0}}^{\text{(n-1) times}},\frac{d}{2^2},\cdots,\overbrace{\mathbf{0},\mathbf{0},\cdots,\mathbf{0}}^{\text{(n-1) times}},\frac{d}{2^l},\cdots \tag{16}$$

Here each word is $m-$bit wide. There are $(n-1)$ zero vectors in each n consecutive output words till the nl^{th} iteration. Note that, with initial states $(d,\mathbf{0},\mathbf{0},\cdots,\mathbf{0})$ with $d=2^lk$, the first ln outputs are same irrespective of any primitive Xorshift RNGs of order n constructed by Alg. 1. So, the initial value of the states of the Xorshift generator produced by the construction algorithm are significant for the quality of pseudorandom vectors generation. To avoid this weakness, the initial states of the Xorshift RNG need be initialized with odd numbers. In such case, all α_i will be equal to 1 at the first iteration and there will be maximum number of active terms for the feedback function computation.

4.2 Different Xorshift generators from same binary primitive polynomial

One of the important thing of the construction algorithm is that it produces different primitive Xorshift generators of different order from a given binary primitive polynomial of degree mn. Since in most of the operating system, the word size is of the form 2^k, we have considered $mn=2^k$ for some positive integer k. Again for $mn=2^k$, there will be $(k-1)$ distinct possible choices for m i.e., $1,2^1,\ldots,2^{k-1}$. For each value of m, the construction algorithm returns n vectors $\{v_0,v_1,\ldots,v_{n-1}\}$, where each v_i is of m-bit length. For better understanding, the following example is provided.

Example 1. Let us consider the binary primitive polynomial $f(x)=x^{32}+x^{31}+x^{27}+x^{26}+x^{25}+x^{20}+x^{19}+x^{15}+x^{14}+x^{11}+x^9+x^7+x^6+x^5+x^4+x^2+1$. Here degree of $f(x)$ is 32 and so $mn=32=2^5$. Then the set of possible choices for m is $\{1,2^1,2^2,2^3,2^4\}$. But, we are only considering $m=2^3$ and 2^4. The respective Xorshift RNGs constructed using Alg. 1 are given below:

1. For $m=2^3$ and $n=2^2$: $v_0=0\text{x}f7$, $v_1=0\text{x}54$, $v_2=0\text{x}73$, $v_3=0\text{x}bf$.
2. For $m=2^4$ and $n=2$: $v_0=0\text{x}bf2f$, $v_1=0\text{x}6775$.

From Eq. (15), it is clear that for a Xorshift generator of order n over the field $\mathbb{F}_{2^m}$ requires following operations in each iteration:

- It requires one right shift operation and at most n XOR operations for computation of the feedback value fd.
- n state shifting operations i.e., $\mathbf{s_i}=\mathbf{s_{i+1}}$ for $i=0,1,\ldots,n-2$ and $\mathbf{s_{n-1}}=fd$.

Then, using Lemma 4, it is clear that at least $(n+2)$ and at most $(2n+1)$ Xorshift operations are needed to produce an m-bit word in each cycle. Thus, to produce a bitstream of length l, it will take $\lceil\frac{l}{m}\rceil$ many iterations. If N is the total number of word operations (XOR, right shift and shifting), then $(n+2)\lceil\frac{l}{m}\rceil\leq N\leq(2n+1)\lceil\frac{l}{m}\rceil$. Suppose for a binary primitive polynomial of degree mn, two separate primitive Xorshift RNGs (RNG1 and RNG2) are generated with word

size m_1 and m_2 respectively, where $m_2 = 2m_1$. Let, the respective order of RNG1 and RNG2 be n_1 and n_2, then $n_1 = \frac{mn}{m_1}$ and $n_2 = \frac{mn}{m_2} = \frac{mn}{2m_1}$. If N_1 and N_2 be the total number of operations required to generate bitstream of length l, then we have

$$(n_1 + 2)\left\lceil \frac{l}{m_1} \right\rceil \leq N_1 \leq (2n_1 + 1)\left\lceil \frac{l}{m_1} \right\rceil, \tag{17}$$

$$(n_2 + 2)\left\lceil \frac{l}{m_2} \right\rceil \leq N_2 \leq (2n_2 + 1)\left\lceil \frac{l}{m_2} \right\rceil. \tag{18}$$

Therefore,

$$\begin{aligned} N_1 &\geq (n_1 + 2)\left\lceil \frac{l}{m_1} \right\rceil \\ &= (2n_2 + 2)\left\lceil \frac{2l}{m_2} \right\rceil \\ &\geq 2(2n_2 + 2)\left(\left\lceil \frac{l}{m_2} \right\rceil - 1\right) \\ &= 2(2n_2 + 1)\left\lceil \frac{l}{m_2} \right\rceil + 2\left\lceil \frac{l}{m_2} \right\rceil \\ &> 2N_2. \end{aligned} \tag{19}$$

Again,

$$N_1 \leq (2n_1 + 1)\left\lceil \frac{l}{m_1} \right\rceil = (4n_2 + 1)\left\lceil \frac{2l}{m_2} \right\rceil \leq 2(4n_2 + 1)\left\lceil \frac{l}{m_2} \right\rceil < 8N_2. \tag{20}$$

Using Eq. (19) and (20), we have $2 < N_1/N_2 < 8$. Therefore, for larger value of m, the Xorshift generator will take lesser number of word operations to produce the bitstream of desired length l and so will take lesser time which is reflected in our experimental results given in the Tab. 2.

Word size m	Avg. Time taken (sec) to generate 10^9 bits	Avg. Time taken (sec) to generate 10^{10} bits
8	78.6	841.1
16	20.0	215.3
32	6.0	62.7
64	1.9	19.2

Table 2 Average timing for different values of m

For our experiment, we have taken $mn = 512$ and the bitstream length l as 10^9 and 10^{10}. Then for $m = 8, 16, 32, 64$, measured the average time taken to generate bitstream of length l. It is observed that if the word size m is increased by 2, then the time taken reduced by c to generate a fixed length bitstream, where $2 < c < 8$. The construction algorithm for primitive Xorshift RNGs is implemented in C and the used Test machine is Intel Xeon(R) CPU E5645 @ 2.40GHz

x 12 with 8 GiB memory and 64-bit Linux operating system. The Tab. 2 summarize the results, which tells that it is better to select the primitive Xorshift RNG having larger word size m (i.e., 32 or 64) so as to take the advantage of modern word based operations.

4.3 Primitive Xorshift generator with Desired Number of Tap Points

The effect of number of tap points in the LFSR (i.e., the number of nonzero coefficients) is important for cryptographic usage while choosing a primitive polynomial. Because an LFSR with less number of tap positions is susceptible to fast correlation attack [16, 17]. The distribution of polynomials over $\mathbb{F}_2$ with respect to their weights are well studied in [18]. It is desirable to select the primitive polynomial whose weight is close to $n/2$ i.e., the polynomial is neither too sparse nor too dense [9, 19]. However, in certain areas like light weight cryptography, it is preferable to have less number of nonzero tap positions. So, it is required to have an algorithm which could generate primitive Xorshift RNGs of order n over $\mathbb{F}_{2^m}$ with desired number of tap points k, where $1 < k < n + 2$. In case of Marsaglia's Xorshift RNGs, there are total six operations (i.e., three XOR and three shifting) are used in the feedback function computation. Alg. 2 produces such primitive RNGs with desired number of Xorshift operations k.

From Lemma 4, it is shown that in case of primitive Xorshift RNGs there will be at least two operations for its feedback computation (i.e., R and $\alpha_0\mathbf{v}_0$). Therefore, for getting a primitive Xorshift generator with k Xorshift operations for its feedback computation, Alg. 2 assigns random binary value to the coefficients needed for the matrix coefficient C_0 as given in Eq. (6) with $a_0 = 1$. Next it selects $(k-2)$ distinct random integer i such that $0 < i < n$ and then constructs the random binary matrix coefficients C_i as described in Eq. (5). Finally, assign $a_{mn} = 1$ so that the polynomial $f(X) = \sum_{i=0}^{mn} a_i X^i$ will be a polynomial of degree mn. If $f(x)$ is primitive, then Alg. 1 returns the desired primitive Xorshift generator.

5 Conclusion

In this paper, we have proposed two algorithms related to Marsaglia's Xorshift RNGs. Alg. 1 constructs primitive Xorshift generator from a given primitive polynomial. We studied those Xorshift generators and found a common weakness in all those generators. It is shown that the states of those Xorshift generators need to be initialized carefully and is suggested that all the states to be initialized with odd numbers. We have shown that several primitive Xorshift generators of different order can be constructed from a given primitive polynomial of degree mn using the construction algorithm. We also shown that for the larger value of word size m, the Xorshift generator takes less time to produce a bitstream of the desired length l. So, in case of software implementations, it is suggested to select the primitive Xorshift generators with a larger word size m (i.e., 32 or 64) to take the advantage of modern word based operations. Finally, We have provided another algorithm that produces efficient primitive Xorshift generator with desired number of Xorshift operations needed for computation of its feedback function .

Acknowledgemnts First author would like to thank to Director, CAIR, DRDO for her encouragement and support, Shri T. S. Raghavan and Dr Subrata Rakshit for their valuable suggestions.

Input: Three positive integers m, n and k.
Output: A primitive Xorshift generator of order n over $\mathbb{F}_{2^m}$ having k Xorshift operations.

Generate a random polynomial $f(X) = \sum_{i=0}^{mn} a_i X^i$ as follows
$i = 1$
while $i < k$ **do**
  if $i = 1$ **then**
    $l = 0$
  else
    $l = rand(\ ) \bmod n$ /*generating random index*/
    while $l \in S$ **do**
      $l = rand(\ ) \bmod n$
    end while
  end if
  $S = S \cup \{l\}$
  $j = 1$
  while $j < m$ **do**
    $a_{l+(j-1)n} = rand(\ ) \bmod 2$ /*generating random bit*/
    $j = j + 1$
  end while
  $i = i + 1$
end while
$a_0 = 1$ and $a_{mn} = 1$
if $f(X)$ is not primitive, go to step-1.
else, using Alg. 1 return the required Xorshift generator from $f(X)$.

Algorithm 2 Primitive Xorshift generator with k Xorshift operations

References

[1] S. W. Golomb. *Shift Register Sequences*. Cambridge University Press, 1967.

[2] R. Lidl and H. Niederreiter. *Finite fields*. Cambridge University Press, 1996. DOI: 10.1017/cbo9780511525926.

[3] S. R. Ghorpade, S. U. Hasan, and M. Kumari. Primitive polynomials, singer cycles and word-oriented linear feedback shift registers. *Designs, Codes and Cryptography*, 58(2):123–134, 2011. DOI: 10.1007/s10623-010-9387-7.

[4] H. Niederreiter. The Multiple-Recursive Matrix Method for Pseudorandom Number Generation. *Finite Fields and Their Applications*, 1(1):3–30, 1995. DOI: 10.1006/ffta.1995.1002.

[5] H. Niederreiter. Pseudorandom vector generation by the multiple-recursive matrix method. *Mathematics of Computation*, 64(209):279–294, 1995. DOI: 10.1090/s0025-5718-1995-1265018-4.

[6] H. Niederreiter. Improved Bounds in the Multiple-Recursive Matrix Method for Pseudorandom Number and Vector Generation. *Finite Fields and Their Applications*, 2(3):225–240, 1996. DOI: 10.1006/ffta.1996.0015.

[7] G. Zeng, W. Han, and K. He. *Word-oriented feedback shift register: σ-LFSR.* Cryptology ePrint Archive: Report 2007/114, 2007.

[8] R. P. Brent. On the periods of generalized Fibonacci recurrences. *Mathematics of Computation*, 63(207):389–389, 1994. DOI: 10.1090/s0025-5718-1994-1216256-7.

[9] F. Panneton and P. L'ecuyer. On the xorshift random number generators. *ACM Transactions on Modeling and Computer Simulation*, 15(4):346–361, 2005. DOI: 10.1145/1113316.1113319.

[10] A. J Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography.* CRC press, 1996.

[11] D. R. Stinson. *Cryptography: theory and practice.* CRC press, 2006.

[12] S. K. Bishoi, H. K. Haran, and S. U. Hasan. A note on the multiple-recursive matrix method for generating pseudorandom vectors. *Discrete Applied Mathematics*, 222:67–75, 2017. DOI: 10.1016/j.dam.2017.01.033.

[13] S. K. Bishoi and V. Matyas. Investigating results and performance of search and construction algorithms for word-based LFSRs, σ-LFSRs. *Discrete Applied Mathematics*, 2018. Accepted for publication.

[14] G. Marsaglia. Xorshift RNGs. *Journal of Statistical Software*, 8(14), 2003. DOI: 10.18637/jss.v008.i14.

[15] D. Knuth et al. *The Art of Computer Programming, Volume 2: Semi numerical Algorithms.* Addison-Wesley Longman, Inc, 1998.

[16] V. Chepyzhov and B. Smeets. On A Fast Correlation Attack on Certain Stream Ciphers. In *Advances in Cryptology — EUROCRYPT '91*, pages 176–185. Springer Berlin Heidelberg. DOI: 10.1007/3-540-46416-6_16.

[17] V. V. Chepyzhov, T. J., and B. Smeets. A Simple Algorithm for Fast Correlation Attacks on Stream Ciphers. In *Fast Software Encryption*, pages 181–195. Springer Berlin Heidelberg, 2001. DOI: 10.1007/3-540-44706-7_13.

[18] P. R. Mishra, I. Gupta, and N. Gaba. Distribution of Primitive Polynomials Over GF(2) with Respect to Their Weights. In *Mathematics and Computing*, pages 441–449. Springer India, 2015. DOI: 10.1007/978-81-322-2452-5_30.

[19] A. Compagner. The hierarchy of correlations in random binary sequences. *Journal of Statistical Physics*, 63(5-6):883–896, 1991. DOI: 10.1007/bf01029989.