Generalized word-oriented feedback shift registers

Susil Kumar Bishoi[1,2], Kedarnath Senapati[2*] and B R Shankar[2]

[1]Center for Artificial Intelligence and Robotics, Defence Research and Development Organisation, CV Raman Nagar, Bengaluru, 560093, Karnataka, India.
[2]Department of Mathematical and Computational Sciences, National Institute of Technology Karnataka, Surathkal, Mangalore, 575025, Karnataka, India.

*Corresponding author(s). E-mail(s): kedar@nitk.edu.in;
Contributing authors: skbishoi@gmail.com; brs@nitk.ac.in;

Abstract

The word-oriented feedback shift registers (WFSRs) possess very attractive properties as they take advantage of modern word-based processors and thus increase the throughput. We provide a generalized form of the feedback function of WFSR along with some special cases. Then, a necessary and sufficient condition for nonsingular WFSR is discussed. We study different word-based cascade systems and the period of sequences produced by these cascade systems is derived. We provide experimental results on avalanche property on states of cascade systems and statistical results of sequences produced by them. Finally, we present a crypt-analytic attack on cascade systems and suggest its countermeasure.

Keywords: Stream cipher, Feedback shift register, Lagged Fibonacci Generator, Multiple-Recursive Matrix Method, Cascade Generator, Linear complexity

MSC Classification: 94A55 , 94A60 , 15B33 , 12E20 and 12E05

1 Introduction

Electronic communications have taken a major key role in the post-pandemic situation as this is contact-less. In the future, its applications will definitely take a huge uptrend. Along with the surge in the use of digital technologies, the rise of online frauds, scams, intrusions, and other security breaches is also increasing. One of the effective solutions for these security breaches is the use of cryptography. Pseudo-random numbers are a major component in cryptography [20, 25] and feedback shift registers (FSRs) have been valuable and practical methods for generating cryptographically secure pseudo-random sequences for cryptographic applications. Linear feedback shift register (LFSR) [13, 18], nonlinear LFSR (NLFSR), multiple-recursive matrix method (MRMM) [2, 3, 27], lagged fibonacci generators (LFG) [5, 9] are among the popular FSRs used in literature. As FSRs have some memory states (resisters), they can not be used in memoryless cipher-like block ciphers, therefore, are usually used in stream ciphers. If FSR word size is 1, then it is known as bit-oriented FSR like LFSR and NLFSR. Mainly, LFSRs are used for keystream generators for stream cipher applications due to their ease and efficiency in implementation. The maximum-length LFSR sequences have many good randomness properties. However, they are not cryptographically secure due to their linear structure. Several methods have been proposed to destroy the linearity of LFSRs, such as combining the outputs of several LFSRs by a nonlinear function, nonlinearly filtering states of an LFSR, and irregularly clocking an LFSR. However, the algebraic attacks and fast algebraic attacks, introduced in [10] and [11], seem to threaten LFSR-based stream ciphers [8]. As an alternative, NFSRs become popular bit-oriented building blocks for stream ciphers. Many of the candidates in the eSTREAM project [12] and CAESAR competition [6] utilize NFSRs as one of their main building blocks.

However, bit-oriented FSRs do not take advantage of modern word-based processors, unlike MRMM and LFG [2, 23, 26]. LFG is a word-oriented FSR with coefficients of either 0 or 1 and produces bitstreams having many good randomness properties. Similarly, MRMM is a word-oriented LFSR with matrices as coefficients, having all statistical properties inherited by LFSR and significantly improving the throughput of sequence generation. Though MRMM takes advantage of the word-based processor and the bitstreams generated by it have good statistical properties, it is vulnerable due to its linear structure. Like in LFSR, several approaches, including the filter generator, nonlinear combination generator, and clock-controlled generators, have been investigated in MRMM to destroy the linear structure. The cascade connection of FSRs with one nonlinear FSR can be used for this purpose. The hardware-oriented finalists Trivium [7] and Grain [16] of eSTREAM project, one of the CAESAR competition finalists ACORN [17] utilize this cascade connection of bit-oriented FSRs. Stream ciphers such as Lizard [14] and Plantlet [21] also use the cascaded type of FSRs. In this paper, we have introduced several word-based cascade systems (CSs) and have studied their periodicity and statistical properties including the avalanche effect.

For all FSRs, the execution steps are similar in each iteration or cycle. It first calculates the feedback value using a feedback function, then shifts the states and stores the feedback value in the last state. In this paper, we have introduced a generalized expression for the feedback function of WFSR and have studied a few particular cases. The remainder of this paper is organized as follows. In Section 2, we introduce some basic concepts and related results. In Section 3, we study WFSR and introduce a generalized form for the feedback function of WFSR and explore nonsingular WFSR. In Section 4 we visit some special cases of WFSRs. Several cascade systems consisting of MRMM and LFG are discussed in Section 5. The final section concludes the paper.

2 Preliminaries

Denote by $\mathbb{F}_2$ and $\mathbb{F}_2^m$ the finite field with two elements and the m-dimensional vector space over $\mathbb{F}_2$, respectively. The symbol $\oplus$ denotes the addition in $\mathbb{F}_2$ and $+$ represents the addition in the residue class ring $\mathbb{Z}/2^m\mathbb{Z}$. The set of all $m \times m$ matrices with entries in $\mathbb{F}_2$ is denoted by $M_m(\mathbb{F}_2)$ and $\mathbb{GL}_m(\mathbb{F}_2)$ represents the set of all $m \times m$ invertible matrices in $M_m(\mathbb{F}_2)$. The $m \times m$ identity matrix is denoted by I_m. For any matrix $C \in M_m(\mathbb{F}_2)$, $\det(C)$ denotes its determinant. Since two finite dimensional vector spaces $\mathbb{F}_{2^m}$ and $\mathbb{F}_2^m$ are isomorphic [22], the element of $\mathbb{F}_{2^m}$ may be thought of as a column vector of size m over $\mathbb{F}_2$ and hence, for any $\mathbf{s} \in \mathbb{F}_{2^m}$ and $C \in M_m(\mathbb{F}_2)$ the matrix-vector multiplication $C\mathbf{s}$ is a well-defined element of $\mathbb{F}_2^m$. We use bold letter variable if it belongs to $\mathbb{F}_{2^m}$ i.e., $\boldsymbol{x} \in \mathbb{F}_{2^m}$, whereas the normal letter x as a bit i.e., $x \in \mathbb{F}_2$. Let $Wt(\mathbf{x})$ denote the hamming weight of $\mathbf{x}$.

We denote by $\mathcal{MP}_m[\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}] = M_m(\mathbb{F}_2)[\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}]/(\boldsymbol{x}_0{}^2 \oplus \boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}{}^2 \oplus \boldsymbol{x}_{n-1})$, the set of all multivariate polynomial $F(.)$ in n variables $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}$ with coefficients in $M_m(\mathbb{F}_2)$ such that $F(\mathbf{0}, \ldots, \mathbf{0}) = \mathbf{0}$ and each $\boldsymbol{x}_i \in \mathbb{F}_{2^m}$. If $F \in \mathcal{MP}_m[\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}]$, then it can be expressed as follows.

$$F(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}) = \sum_{I \in P(N)} C_I \prod_{k \in I} \boldsymbol{x}_k \tag{1}$$

where $P(N)$ denotes the power set of $N = \{0, \ldots, n-1\}$ and $C_I \in M_m(\mathbb{F}_2)$. For example, if $N = \{0, 1, 2\}$ then the general expression of $F(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2) = C_0\boldsymbol{x}_0 + C_1\boldsymbol{x}_1 + C_2\boldsymbol{x}_2 + C_{01}\boldsymbol{x}_0\boldsymbol{x}_1 + C_{02}\boldsymbol{x}_0\boldsymbol{x}_2 + C_{12}\boldsymbol{x}_1\boldsymbol{x}_2 + C_{012}\boldsymbol{x}_0\boldsymbol{x}_1\boldsymbol{x}_2$. In Eq. (1), $\mathbf{X} = \prod_{k \in I} \boldsymbol{x}_k$ is computed first, which returns an m-bit number and then $C_I\boldsymbol{X}$ is calculated using matrix-vector multiplication. Here the product $\prod$ can be either field multiplication or modular integer multiplication $*$ or bitwise AND operation &. Similarly, the sum $\sum$ is either modular integer addition $+$ or bitwise XOR operation $\oplus$. As field multiplication is an expensive operation compared to the other two, so we only focus on the other two multiplication operations $*$ and &. The algebraic degree of F denoted as $deg(\mathrm{F})$ can be defined as $\max\{|I| : C_I \neq 0\}$, where $|I|$ denotes the size of I.

3 Word-oriented FSR

An n-stage word-oriented FSR (WFSR) is an FSR where each stage stores a word of size m-bits. A state of a WFSR is a vector $(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1})$, where $\boldsymbol{x}_i$ indicates the content of stage i. Let $\mathbf{S}_0 = (\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{n-1})$ be the initial state of WFSR and suppose $F \in \mathcal{MP}_m[\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}]$ is the feedback function for WFSR. At every clock pulse, there is a transition from the state $\mathbf{S}_t = (\mathbf{s}_t, \mathbf{s}_{t+1}, \ldots, \mathbf{s}_{t+n-1})$ to the state $\mathbf{S}_{t+1} = (\mathbf{s}_{t+1}, \mathbf{s}_{t+2}, \ldots, \mathbf{s}_{t+n})$ for some integer $t \geq 0$, where $\mathbf{s}_{n+t} = F(\mathbf{s}_t, \mathbf{s}_{t+1}, \ldots, \mathbf{s}_{t+n-1})$. After consecutive clock pulses, the WFSR outputs a word sequence $[\mathbf{s}] = \{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_n, \ldots\}$. The sequence $[\mathbf{s}]$ is *ultimately periodic* if there are integers r, n_0 with $r \geq 1$ and $n_0 \geq 0$ such that $\mathbf{s}_{j+r} = \mathbf{s}_j$ for all $j \geq n_0$. If $n_0 = 0$, then $[\mathbf{s}]$ is said to be periodic and in such case, the least positive integer r is called the *period* of the sequence $[\mathbf{s}]$. A WFSR is called a linear WFSR if its feedback function F is linear and a nonlinear WFSR otherwise.

Before discussing WFSR further, we present three WFSRs in the following example.

Example 1 Consider a three-stage WFSR with word size 2 called WFSR_1. The feedback functions of WFSR_1 is considered as $F_1(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2) = C_0\boldsymbol{x}_0 + C_{12}\boldsymbol{x}_1\boldsymbol{x}_2$ where $C_0 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ and $C_{12} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Here modular integer addition and modular integer multiplication are used in the feedback value computation. Using the feedback function, one can compute the next state of a given state. The number of all possible states for a WFSR of length 3 with word size 2 is 4^3. Fig. 1 shows a part of the state diagram of WFSR_1 with an initial state $(1, 2, 3)$. For abuse of notation, we use 123 for the state $(1, 2, 3)$. Here each number is converted to a vector and vice-versa during the feedback value calculation. For this example, $0 = [0, 0]^t, 1 = [0, 1]^t, 2 = [1, 0]^t$ and $3 = [1, 1]^t$ are used. It is easy to verify that the next state of 123 is 231 as $F(1, 2, 3) = 1$. In this case, the state 231 is called the successor of 123, and 123 is called the predecessor of 231.

Consider WFSR_2, another three-stage WFSR with word size 2 with feedback function $F_2(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2) = C_0\boldsymbol{x}_0 + C_{12}\boldsymbol{x}_0\boldsymbol{x}_1\boldsymbol{x}_2$ where the matrix coefficients C_0 and C_{12} are same as in WFSR_1. Fig. 2 shows a part of the state diagram of WFSR_2 with the same initial state 123. Suppose WFSR_3 is another three-stage WFSR with word size 2 having feedback function $F_3(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2) = C_0\boldsymbol{x}_0 + C_{12}(\boldsymbol{x}_1 \& \boldsymbol{x}_2)$. The feedback function is the same as in WFSR_1 except the operation & is used in place of $*$. Fig. 3 shows a part of the state diagram of WFSR_3 with the same initial state 123.

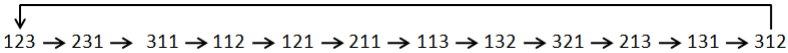

Fig. 1 The part of state diagrams of WFSR_1 with initial state 123

Unlike WFSR_1 and WFSR_3, distinct vectors do not have distinct successors in WFSR_2. The states 233 and 330 of WFSR_2 have common successors 301 i.e.,

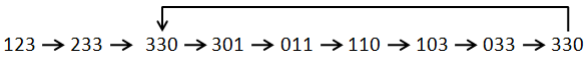

Fig. 2 The part of state diagrams of WFSR_2 with initial state 123

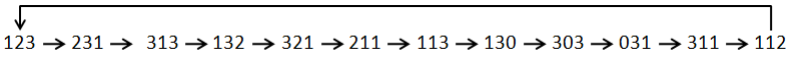

Fig. 3 The part of state diagrams of WFSR_3 with initial state 123

301 does not have unique predecessors. A WFSR (or its feedback function) is called nonsingular if each state has a unique predecessor i.e., its state diagram consists of disjoint cycles. In the case of bit-oriented FSR, it is shown in [13] that the FSR is nonsingular if and only if its feedback function is of the form

$$f(x_0, x_1, \ldots, x_{n-1}) = x_0 \oplus g(x_1, x_2, \ldots, x_{n-1}) \tag{2}$$

where the function $g(.)$ does not depend on the variable x_0. Now the natural question arises, is there any necessary and sufficient condition for WFSR to be nonsingular? It is easy to show that the WFSR is nonsingular if its feedback function is of the form as in Eq. (2). There are other WFSRs whose feedback functions are in different form as WFSR_1. In the following, we provide a necessary and sufficient condition for which WFSR is nonsingular.

Theorem 1 *An n-stage WFSR is nonsingular if and only if its feedback function* $F \in \mathcal{MP}_m[\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}]$ *can be represented as*

$$F(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}) = f_0(\boldsymbol{x}_0) \odot f_1(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}) \tag{3}$$

where f_0 *is a bijective function and* f_1 *is an arbitrary function* $\mathbb{F}_{2^m}^{n-1} \to \mathbb{F}_{2^m}$. *The operation* $\odot$ *is either* $+$ *or* $\oplus$.

Proof Suppose WFSR is nonsingular, then distinct vectors have distinct successors. If the feedback function F does not contain any $\boldsymbol{x}_0$ term, then it is easy to get two distinct vectors having a common successor. Thus, the expression of F must contains $\boldsymbol{x}_0$ term and so $F(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1})$ can be expressed as $f_0(\boldsymbol{x}_0) \odot \boldsymbol{x}_0 f_2(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}) \odot f_1(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1})$ where $f_0(\boldsymbol{x}_0)$ contains all terms of single variable $\boldsymbol{x}_0$ and both the functions f_1 and f_2 are independent of $\boldsymbol{x}_0$. Also f_2 does not have any constant term as all $\boldsymbol{x}_0$ terms are in f_0 and so $f_2(0, \ldots, 0) = 0$. If f_0 is not bijective, then there exists $\boldsymbol{x}_0^{'} \neq \boldsymbol{x}_0^{''}$ such that $f_0(\boldsymbol{x}_0^{'}) = f_0(\boldsymbol{x}_0^{''})$. This implies that $F(\boldsymbol{x}_0^{'}, 0, \ldots, 0) = F(\boldsymbol{x}_0^{''}, 0, \ldots, 0)$ and thus $(\boldsymbol{x}_0^{'}, 0, \ldots, 0)$ and $(\boldsymbol{x}_0^{''}, 0, \ldots, 0)$ have common successor. This is a contradiction and so f_0 is bijective. Next to prove that $f_2 = 0$. Suppose f_2 has some nonzero terms, then $f_2(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1})$ must contain a term of $\boldsymbol{x}_k$ for $k \neq 0$. In this case, $F(0, 2, \ldots, 2) = F(2^{m-1}, 2, \ldots, 2)$. This is not possible as WFSR is nonsingular and so $f_2(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}) = 0$. This proves the necessary part.

To prove WFSR is nonsingular, we need to show distinct vectors have distinct successors. If the two vectors $(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1})$ and $(\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n-1})$ differ in any component other than the first, then their successors $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$ and $(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n)$ are still distinct. Thus, WFSR is nonsingular if and only if

$(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1})$ and $(\boldsymbol{y}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1})$ have distinct successors for $\boldsymbol{x}_0 \neq \boldsymbol{y}_0$. Suppose WFSR is singular, then there exists $\boldsymbol{x}_0 \neq \boldsymbol{y}_0$ such that $f(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}) = f(\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n-1})$. This implies that $f_0(\mathbf{x}_0) = f_0(\mathbf{y}_0)$ for $\boldsymbol{x}_0 \neq \boldsymbol{y}_0$. This is a contradiction as f_0 is bijective. This completes the proof.

□

Corollary 2 Bit-oriented FSRs are nonsingular if and only if the feedback function $f(x_0, x_1, \ldots, x_{n-1}) = x_0 \oplus g(x_1, x_2, \ldots, x_{n-1})$.

Proof In the case of bit-oriented FSR, $m = 1$. Thus, there are two bijective functions from $\mathbb{F}_2 \longrightarrow \mathbb{F}_2$. In both cases, the feedback function f is expressed in the desired form. □

Note that if $deg(F) = 1$, then the feedback function F in Eq. (1) satisfies the criteria of Theorem 1 and so the feedback function is always nonsingular.

4 Some special cases of WFSR

4.1 Bit-oriented FSRs

The WFSR becomes a bit-oriented FSR when $m = 1$. In this case, the $m \times m$ matrix coefficient C_k of Eq. (1) becomes a scalar $c_k \in \mathbb{F}_2$. If $deg(F) = 1$, then $F(x_0, \ldots, x_{n-1}) = \bigoplus_{k=0}^{n-1} c_k x_k$. This expression is the feedback function of well known FSR called LFSR [13] and the theory of LFSRs is well-developed. If the feedback polynomial of LFSR is primitive, then it generates maximal periodic bitstreams for any nonzero initialization of LFSR states and these bitstreams have most of the statistical properties.

If $deg(F) > 1$ then bit-oriented FSR becomes NLFSR. There is no efficient way to find a feedback function such that its corresponding NFSR can generate bitstream with guaranteed long periods. On the other hand, given a feedback function, it is hard to predict the periods of NFSR sequences. However, Golomb proved that all sequences generated by an NFSR are periodic if and only if its feedback function is nonsingular. The maximum period that can achieve by an n-stage NFSR is 2^n and in this case it is known as de Bruijn sequences. Unlike the well-developed theory of LFSRs, the theory of NFSRs is not well-understood due to its complexity.

4.2 MRMM

If $deg(F) = 1$, word size $m > 1$ and $\oplus$ is used in place of $\odot$, then the feedback function as expressed in Eq. (1) becomes $F(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1}) = \bigoplus_{k=0}^{n-1} C_k \boldsymbol{x}_k$ where $C_0, C_1, \ldots, C_{n-1} \in M_m(\mathbb{F}_2)$. This is the general expression for the feedback

function of MRMM [22]. For a periodic word sequence [**s**], it is always possible to have a relation called linear recurring relation (LRR) among the elements as

$$\mathbf{s}_{i+n} = \sum_{k=0}^{n-1} C_k \mathbf{s}_{i+k} \quad i \geq 0. \tag{4}$$

where $C_0, C_1, \ldots, C_{n-1} \in M_m(\mathbb{F}_2)$ [22]. The Eq. (4) for [**s**] can be mapped to an MRMM of order n over $\mathbb{F}_{2^m}$ which we denote as MRMM(m, n). The sequence [**s**] is referred as the sequence generated by the MRMM(m, n) and the polynomial associated with Eq. (4) denoted as $M(x) = I_m x^n - C_{n-1}x^{n-1} - \cdots - C_1 x - C_0$ with matrix coefficients is called the *matrix polynomial* of the MRMM(m, n). The following proposition from [15] tells some basic facts about MRMMs.

Proposition 3 *For the sequence* [**s**] *generated by the MRMM*(m, n)*, we have*

(i) [**s**] *is ultimately periodic and its period is no more than* $2^{mn} - 1$.
(ii) *if* C_0 *is invertible, then* [**s**] *is periodic. Conversely, if* [**s**] *is periodic whenever the initial state is of the form* $(\mathbf{b}, 0, \ldots, 0)$*, where* $\mathbf{b} \in \mathbb{F}_{2^m}$ *with* $\mathbf{b} \neq 0$*, then* C_0 *is invertible.*

An MRMM(m, n) is called primitive if, for any choice of the nonzero initial state, the sequence generated by that MRMM is periodic with period $2^{mn}-1$. It is shown in [2] that the MRMM(m, n) is primitive if and only if the determinant of the matrix polynomial $M(x)$ of MRMM(m, n) is a primitive polynomial of degree mn over $\mathbb{F}_2$.

4.3 LFG

If $deg(F) = 1$, word size $m > 1$ and $+$ is used in place of $\odot$, then $F(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1}) = C_0\boldsymbol{x}_0 + C_1\boldsymbol{x}_1 + \ldots + C_{n-1}\boldsymbol{x}_{n-1}$. If all $C_k \in \{I_m, 0\}$, then $F(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1})$ is the general expression of the feedback function of an additive LFG. If $f(x) = x^n - c_{n-1}x^{n-1} - \ldots - c_1 x - c_0$ is the corresponding characteristic primitive polynomial of LFG, then it is proved by R. P. Brent [5] that the period of the recurrence relation is $2^{m-1}(2^n - 1)$ for all $m \geq 1$ if and only if $f(x)^2 + f(-x)^2 \neq 2f(x^2)$ (modulo 8) and $f(x)^2 + f(-x)^2 \neq 2(-1)^n f(-x^2)$ (modulo 8). Using matrix theory, George Marsaglia et al. [19] also showed in their paper that the recurrence relation has the maximal period $(2^n - 1)2^{m-1}$ for all $m \geq 1$ and for every initial state with at least one odd number if and only if the transition matrix A corresponding to the characteristic primitive polynomial has the following three properties:

- order $j = 2^n - 1$ in the group of nonsingular matrices for mod 2,
- order $2j$ for mod 2^2,
- order $4j$ for mod 2^3.

When an LFG has a maximal period, it is known as a primitive LFG.

5 Cascade connection of WFSRs

The word-oriented cascade connection of FSRs was discussed in [4], where the cascade connection of MRMMs was analyzed. For the sake of completeness, we revisit the word-oriented cascade connection again. In this paper, we focus on a cascade system (CS) comprises of two WFSRs only. Suppose WFSR_1 and WFSR_2 are those two WFSRs where WFSR_1 is cascaded into WFSR_2 as depicted in Fig. 4.

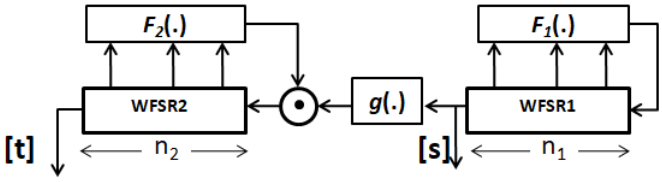

Fig. 4 The cascade connection of WFSR_1 into WFSR_2.

Consider the orders of WFSR_1 and WFSR_2 be n_1 and n_2, respectively. Let $\mathbf{S}_0 = \{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{n_1-1}\}$ and $\mathbf{T}_0 = \{\mathbf{t}_0, \mathbf{t}_1, \ldots, \mathbf{t}_{n_2-1}\}$ be the initial states of WFSR_1 and WFSR_2. Suppose $F_1(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_1-1})$ is the feedback function for WFSR_1, mapping from $\mathbb{F}_{2^m}^{n_1}$ to $\mathbb{F}_{2^m}$. Similarly, let $F_2 : \mathbb{F}_{2^m}^{n_2} \to \mathbb{F}_{2^m}$ be the feedback function for WFSR_2. Let $g : \mathbb{F}_{2^m} \to \mathbb{F}_{2^m}$ be a bijective function. Consider the word sequence $[\mathbf{s}]$ is generated by WFSR_1 in free running mode and $[\mathbf{t}]$ is generated by the cascade connection of WFSR_1 into WFSR_2. We say WFSR_2 is running in scrambler mode and the feedback value of WFSR_2 is modified as follows

$$\mathbf{t}_{i+n_2} = g(\mathbf{s}_i) \odot F_2(\mathbf{t}_i, \mathbf{t}_{i+1}, \ldots, \mathbf{t}_{i+n_2-1}), \ \textit{for } i \geq 0, \tag{5}$$

where the operation $\odot$ can be either $\oplus$ or modular addition. Let P_s and P_t be the period of the sequences $[\mathbf{s}]$ and $[\mathbf{t}]$, respectively. Then we have the following relation between P_s and P_t.

Theorem 4 *If WFSR_1 is periodic and WFSR_2 is nonsingular, then $[\mathbf{t}]$ is periodic and P_s divides P_t.*

Proof It is obvious that the sequence $[\mathbf{t}]$ will be ultimately periodic and let Q_t be the preperiod of $[\mathbf{t}]$. Then, $\mathbf{t}_i = \mathbf{t}_{i+P_t}$ for any $i \geq Q_t$ and so $\mathbf{t}_{i+n_2} = \mathbf{t}_{i+n_2+P_t}$ for any $i \geq Q_t$. So by Eq. (5), $g(\mathbf{s}_i) + F_2(\mathbf{t}_i, \ldots, \mathbf{t}_{i+n_2-1}) = g(\mathbf{s}_{i+P_t}) + F_2(\mathbf{t}_{i+P_t}, \ldots, \mathbf{t}_{i+P_t+n_2-1})$. But P_t is the period of $[\mathbf{t}]$ and thus, $g(\mathbf{s}_i) = g(\mathbf{s}_{i+P_t})$ for all $i \geq Q_t$. As g is bijective, $\mathbf{s}_i = \mathbf{s}_{i+P_t}$ for $i \geq Q_t$. This implies that $\mathbf{s}_i = \mathbf{s}_{i+P_t}$ for any $i \geq 0$ as $[\mathbf{s}]$ is periodic. But, the period of $[\mathbf{s}]$ is P_s and so P_s divides P_t.

Suppose, $[\mathbf{t}]$ is not periodic i.e., $Q_t > 0$. Then, $\mathbf{t}_{Q_t+n_2-1} = \mathbf{t}_{Q_t+n_2-1+P_t}$ as P_t is the period of $[\mathbf{t}]$ and $Q_t + n_2 - 1 \geq Q_t$. This implies, $g(\mathbf{s}_{Q_t-1}) \odot F_2(\mathbf{t}_{Q_t-1}, \ldots, \mathbf{t}_{Q_t+n_2-2}) = g(\mathbf{s}_{Q_t-1+P_t}) \odot F_2(\mathbf{t}_{Q_t-1+P_t}, \ldots, \mathbf{t}_{Q_t+n_2-2+P_t})$. Since P_s divides P_t, it implies $\mathbf{s}_{Q_t-1} = \mathbf{s}_{Q_t-1+P_t}$ and so $g(\mathbf{s}_{Q_t-1}) = g(\mathbf{s}_{Q_t-1+P_t})$. Thus, $F_2(\mathbf{t}_{Q_t-1}, \ldots, \mathbf{t}_{Q_t+n_2-2}) = F_2(\mathbf{t}_{Q_t-1+P_t}, \ldots, \mathbf{t}_{Q_t+n_2-2+P_t})$. Again FSR2 is non-singular and so by Theorem 1, F_2 can be expressed as $F_2(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1}) = f_{20}(\mathbf{x}_0) \odot f_{21}(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1})$ with f_{20} is one-one. This results $f_{20}(\mathbf{t}_{Q_t-1}) = f_{20}(\mathbf{t}_{Q_t-1+P_t})$ and so $\mathbf{t}_{Q_t-1} = \mathbf{t}_{Q_t-1+P_t}$. Thus, $\mathbf{t}_i = \mathbf{t}_{i+P_t}$ for any $i \geq Q_t - 1$. This is a contradiction to the fact that Q_t is the preperiod of $[\mathbf{t}]$. Hence $Q_t = 0$ and this completes the proof. □

With $\odot$ as $\oplus$ and the function $g(.)$ as the identity function, the above result is proved in [4, Theorem 9]. In the following section, we study the statistical properties of the bitstream generated by different CSs using different values of $\odot$ and later analyze the avalanche effect on states of CSs.

5.1 Cascade connection of two MRMMs

In this section, we analyze the randomness properties of the bitstreams generated by three MRMM-based CSs. We call CS as CS_1 (or CS_2) when $\oplus$ (or $+$) is used for $\odot$ in Eq.(5). The CS_1 and CS_2 are depicted in Fig. 5 and Fig. 6. In both cases, g is taken as an identity map. The third cascade system CS_3 is the same as CS_1 with $g(.)$ as a nonlinear bijection function S box. The S box is given in Appendix A. For experimental study in each CS, both FSRs are taken primitive MRMMs defined over $\mathbb{F}_{2^{16}}$ i.e., $m = 16$. The order of $MRMM_1$ and $MRMM_2$ is 10 and 7, respectively. The matrix polynomial of $MRMM_1$ and $MRMM_2$ are $x^{10} + R_6x^8 + LCS_8x^7 + R_8x^6 + 62924x^5 + R_9x^4 + R_4x^1 + LCS_1$ and $x^7 + R_{11}x^4 + LCS_8x^3 + L_9x^2 + R_5x^1 + LCS_{14}$, respectively i.e., $n_1 = 10$ and $n_2 = 7$. Here R_a is the right shift operator defined as $R_a(X) = X >> a$ whereas L_b is the left shift operator defined as $L_b(X) = X << b$ and the left circular shift of X by c-bit is denoted as $LCS_c(X)$. As $MRMM_1$ runs in free running mode, we avoid all zero-state initialization situations in $MRMM_1$. It is observed that the results of the statistical test suite are similar for several random initializations of the states of $MRMM_2$ including all zero initialization states. This occurs as the states of $MRMM_1$ are injected into the states of $MRMM_2$ through its feedback calculation. Thus, the states of $MRMM_2$ can be initialized with any random values, but for our experimental set up we always initialize with zero values.

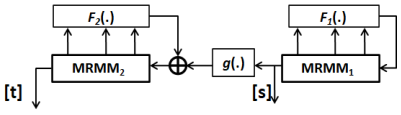

Fig. 5 CS_1

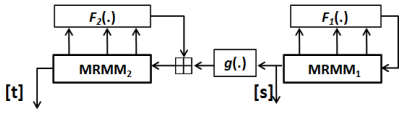

Fig. 6 CS_2

5.1.1 Comparison of statistical properties of CS_1, CS_2 and CS_3

Many statistical test suites have been developed to study and analyze the randomness properties of a bitstream. In our experiment, we use NIST statistical test suite SP 800-22 Rev.1a [1]. To see the randomness properties of the word sequences generated by CS_1 and CS_2, we first convert word sequence $[\mathbf{t}]$ to bit sequence. By Theorem 4, $(2^{160}-1)$ is a factor of the period of the word sequence, so the period of the corresponding bit sequence will be at least $16(2^{160}-1)$ as each word is 16-bit wide. We investigate the randomness properties of the bitstream of CS_1 as follows:

1. Initialize of states $\mathbf{S}_0$ and $\mathbf{T}_0$: Initialize all the states of $MRMM_1$ and $MRMM_2$ to $\mathbf{0}$ except $\mathbf{s}_0 = 1$ i.e., $\mathbf{s}_k = \mathbf{0}$ for $0 < k \leq n_1 - 1$ and $\mathbf{t}_k = \mathbf{0}$ for $0 \leq k \leq n_2 - 1$. Run for 48 iterations without collecting any bitstream. The reason for the selection of 48 rounds is explained in Section 5.1.2.
2. Then for each iteration end, collect 16-bit keystream as $\mathbf{t}_0$. In this experiment, it generates 100 bitstream files and each file consists of a bitstream of length 1000000. For this, CS_1 runs 6250048 times to generate these 100 files.
3. Then, NIST statistical test suite is performed on all those 100 bitstream files. For this experiment, the significant value α is taken 0.05.

Similar to CS_1, 100 bitstream files are generated for CS_2 and CS_3, then NIST statistical test suite is performed. Note that both CS_1 and CS_2 are the same except for different operations $\odot$ used in Eq.(5). The comparative statistical results are provided in Table 1. The first column tells the name of each statistical test of the NIST test suite. The second, third, and fourth columns show the number of passed files out of 100 in the case of CS_1, CS_2, and CS_3, respectively.

As $[\mathbf{s}]$ is generated by a primitive MRMM, it has good randomness properties. But it has low linear complexity due to linear structure. These randomness properties will be inherited by CS_1 as $[\mathbf{s}]$ is xored in the generation of $[\mathbf{t}]$. This fact is reflected in the second column of Table 1. It is checked that the linear complexity of all files of CS_1 is $4352 = m^2(n_1 + n_2)$ as expected. Thus, none of the bitstreams of length 1000000 passes the linear complexity test as the expected linear complexity is 500000. Table 1 shows that statistical results on the bitstream of CS_1 and CS_2 are very close except for the linear complexity. In the case of the bitstream of CS_2, it is tested that linear complexity is around 500000 for all 100 bitstreams of length 1000000. This is due to use of one modular addition[1]. Due to the presence of a nonlinear S box in CS_3, the bitstreams of CS_3 have also good statistical properties like CS_2. Therefore, both CS_2 and CS_3 have an advantage over CS_1 with respect to randomness property. The S

[1]It is known that if $a = \sum_{i=0}^{m-1} a_i 2^i$, $b = \sum_{i=0}^{m-1} b_i 2^i$ and $c = a + b = \sum_{i=0}^{m-1} c_i 2^i$, with $a_i, b_i, c_i \in \{0,1\}$, then $c_0 = a_0 \oplus b_0$, and for $1 \leq k < m$, $c_k = a_k \oplus b_k \oplus \sum_{i=0}^{k-1} a_i b_i \prod_{j=i+1}^{k-1} (a_j \oplus b_j)$. As the value of word size m is 16, the nonlinear degree of the expression of c_{15} will be 16.

Name of the test	No. of passed files CS_1	No. of passed test CS_2	No. of passed test CS_3
Frequency	95	95	97
BlockFrequency	96	99	99
Runs	98	94	97
LongestRunOfOnes	98	97	94
Rank	93	94	92
DiscreteFourierTransform	92	90	94
NonOverlappingTemplateMatchings	100	100	100
OverlappingTemplateMatchings	93	95	94
Universal	94	97	94
LinearComplexity	0	100	100
Serial	98	94	97
ApproximateEntropy	97	93	95
CumulativeSums	95	96	96
RandomExcursions	60	59	64
RandomExcursionsVariant	60	58	62

Table 1 Comparative statistical results of CS_1 and CS_2

box used in CS_3 is generated by the S box construction method applied in the AES block cipher.

5.1.2 Avalanche analysis on state registers of CS_1, CS_2 and CS_3

In this section, we analyze the avalanche effect on the states of three CSs. For the avalanche analysis study, we kept the same MRMMs used in the previous section for statistical analysis. After initialization of $\mathbf{S}_0$ and $\mathbf{T}_0$, we compute experimentally the number of iterations needed to achieve avalanche property, i.e., 50% bit changes in the states of each of three CSs separately. It is observed that number of iterations to achieve 50% avalanche in the case of CS_1 and CS_2 is very close for any random initialization of $\mathbf{S}_0$ and $\mathbf{T}_0$. However, depending upon the total weight of the initial states of CS, the number of iterations varies to achieve 50% avalanche effect as shown in Table 2 and Table 3. The layout of the experiment to generate Table 2 is as follows.

1. Set $count = 0$.
2. Initialize the states of $MRMM_1$ and $MRMM_2$: $\mathbf{s}_k = \mathbf{ini\text{-}s}_k = \mathbf{0}$ for $0 < k \leq n_1 - 1$ and $\mathbf{s}_0 = \mathbf{ini\text{-}s}_0 = 1$. Here $\mathbf{ini\text{-}s}_k$ is the initial state of $\mathbf{s}_k$. Initialize $\mathbf{t}_k = \mathbf{ini\text{-}s}_k = \mathbf{0}$ for $0 \leq k \leq n_2 - 1$
3. Run $MRMM_1$ for one iteration. Here one iteration means it calculates the feedback value fb_1 of $MRMM_1$, then shift the states i.e., $\mathbf{s}_k = \mathbf{s}_{k+1}$ for $0 \leq k < n_1 - 1$ and $\mathbf{s}_{n_1-1} = fb_1$.
4. Run $MRMM_2$ for one iteration i.e., it calculates the feedback value fb_2 of $MRMM_2$, then shift the states i.e., $\mathbf{t}_k = \mathbf{t}_{k+1}$ for $0 \leq k < n_2 - 1$ and $\mathbf{t}_{n_2-1} = fb_2 \odot \mathbf{s}_0$.
5. Calculate $weight = \sum_{k=0}^{n_1-1} Wt(\mathbf{s}_k \oplus \mathbf{ini\text{-}s}_k) + \sum_{k=0}^{n_2-1} Wt(\mathbf{t}_k \oplus \mathbf{ini\text{-}t}_k)$.

6. If $|\ weight - \frac{m(n_1+n_2)}{2}\ | \geq 3$, increase *count* by 1 and go to step-3. Otherwise, return *count*. Here 3 is the approximation value of 1% of the total size of memory states i.e., $m(n_1 + n_2)$.

Table 2 Avalanche effects Vs iteration numbers when all states except $\mathbf{s}_0$ are **0**

	CS_1		CS_2		CS_3	
Iter. No.	Wt. of states	avalanche effect %	Wt. of states	avalanche effect %	Wt. of states	avalanche effect %
0	3	1.10	3	1.10	9	3.31
1	3	1.10	3	1.10	17	6.25
5	7	2.57	7	2.57	47	17.28
10	18	6.62	18	6.62	66	24.26
15	35	12.87	36	13.24	71	26.10
20	62	22.79	62	22.79	94	34.56
30	119	43.75	120	44.12	123	45.22
40	124	45.59	120	44.12	124	45.59

Table 3 Avalanche effects Vs iteration numbers when all states of $MRMM_1$ are 0xFFFF and all states of $MRMM_2$ are **0**

	CS_1		CS_2		CS_3	
Iter. No.	Wt. of states	avalanche effect %	Wt. of states	avalanche effect %	Wt. of states	avalanche effect %
0	23	8.46	23	8.46	17	6.25
1	46	16.91	46	16.91	34	12.50
2	68	25.00	68	25.00	50	18.38
3	90	33.09	83	30.51	68	25.00
4	104	38.24	107	39.34	80	29.41
5	125	45.96	124	45.59	97	35.66
6	144	52.94	144	52.94	115	42.28

Table 2 shows the total weight of the states and the percentage of total states size of CS_1, CS_2, and CS_3, respectively with respect to the iteration number. Here the percentage of avalanche effect is calculated as (total weight of the states of CS)/(total size of CS in bits) $\times$ 100. In this case, the total size of CS in bits is 272. It is observed experimentally that both CS_1 and CS_2 take almost the same number of iterations to achieve 50% avalanche effect for the same nonzero random initialization of states.

Table 3 tells that around 6 iterations are needed to achieve the desired effect. In this case, data are generated after all states of $MRMM_1$ are initialized with 0xFFFF i.e., all bits of $MRMM_1$ are 1, and all states of $MRMM_2$ are kept 0. It is also noticed that if the states of $MRMM_1$ and $MRMM_2$ are balanced, then CS takes around 9 iterations to achieve 50% avalanche effect. Except $\mathbf{s}_0$, if all states of CS_1 and CS_2 are in zero states, both CS_1 and CS_2 achieve the avalanche property after 48 iterations. Therefore, during the study of statistical properties of CS_1 and CS_2 in Section 5.1.1 we have collected bitstream after 48 iterations.

5.2 Cascade connection of MRMM and LFG

In the previous section, we studied cascade systems comprised of MRMMs only. In this section, we focus on two more cascade systems. One is CS_4 consists of two LFGs and the other is CS5 where one FSR is MRMM and the other is LFG. Consider both the LFGs used in CS_4 are two primitive additive LFGs where LFG_1 is cascaded into LFG_2. If $[\mathbf{t}]$ is a nonzero sequence generated by this cascade system, then we have the following result.

Theorem 5 *Let the order of LFG_1 and LFG_2 be n_1 and n_2, respectively having common word size m. If $gcd(n_1, n_2) = 1$, then $Per[\boldsymbol{t}] = (2^{n_1} - 1)(2^{n_2} - 1)2^{m-1}$.*

Proof Suppose $[\mathbf{s}]$ is the word sequence generated by LFG_1. Then $[\mathbf{s}]$ is periodic as LFG_1 is primitive and $Per[\mathbf{s}] = (2^{n_1} - 1)2^{m-1}$. Again the feedback function $f_2(.)$ of LFG_2 is nonsingular and so $(2^{n_1} - 1)2^{m-1}$ divides $Per[\mathbf{t}]$ by Theorem (4). Let $[\mathbf{t}^{(1)}]$ be the bit sequence consisting of the first least significant bit (LSB) of each word of $[\mathbf{t}]$. Then, it can be visualized that $[\mathbf{t}^{(1)}]$ is generated by the cascade connection of two primitive LFSRs. Thus $Per[\mathbf{t}^{(1)}] = (2^{n_1} - 1)(2^{n_2} - 1)$ by [4, Corollary 14]. This implies $(2^{n_1} - 1)(2^{n_2} - 1)2^{m-1}$ divides $Per[\mathbf{t}]$ as $Per[\mathbf{t}^{(1)}]$ divides $Per[\mathbf{t}]$.

Again, it is obvious that $Per[\mathbf{t}]$ divides $lcm((2^{n_1} - 1)2^{m-1}, (2^{n_2} - 1)2^{m-1})$ as both the LFGs are primitive. As $gcd(n_1, n_2) = 1$, $lcm((2^{n_1} - 1)2^{m-1}, (2^{n_2} - 1)2^{m-1}) = (2^{n_1} - 1)(2^{n_2} - 1)2^{m-1}$. This proves the desired result. □

Since the period of CS_4 is $(2^{n_1} - 1)(2^{n_2} - 1)2^{m-1}$, the lower bound for linear complexity of any nonzero bitstream of CS_4 is $n_1 n_2 2^{m-2}$. In CS5 LFG cascades into MRMM. Then using similar arguments, the following theorem can be proved.

Theorem 6 *Let the order of LFG and MRMM be n_1 and n_2, respectively having common word size m. If $gcd(n_1, mn_2) = 1$, then $Per[\boldsymbol{t}] = (2^{n_1} - 1)(2^{mn_2} - 1)2^{m-1}$.*

5.3 Cryptanalysis of word-based CSs

In the previous section, we studied cascade systems comprise of MRMMs and LFGs, then some of their statistical properties. In this section, we discuss a cryptanalytic attack and its complexity. We start with a general method for reconstructing the original sequence from a known portion of the output word sequence. Assume that $[\mathbf{t}]$ is the known output word sequence of the cascade system of Eq. (5) where the function $g(.)$ is the Identity function. Thus, $\mathbf{t}_{i+n_2} = \mathbf{s}_i \odot F_2(\mathbf{t}_i, \mathbf{t}_{i+1}, \ldots, \mathbf{t}_{i+n_2-1})$, *for* $i \geq 0$. Our aim is to reconstruct the initial states of both the FSRs (i.e., $\mathbf{s}_0, \ldots, \mathbf{s}_{n_1-1}$ and $\mathbf{t}_0, \ldots, \mathbf{t}_{n_2-1}$). Since the procedure of cryptanalytic attack is similar for all discussed cascade systems comprised of MRMMs and LFGs except CS_3, we only focus on CSs of LFGs i.e., CS_4. Consider LFG_1 is cascaded into LFG_2 in the CS and both LFGs are primitive. Let $[\mathbf{t}^{(1)}]$ be the bit sequence consisting of the first LSB of each

word of $[\mathbf{t}]$. Then, $[\mathbf{t}^{(1)}]$ is a bit sequence generated by a cascade system of two primitive LFSRs. If n_1 and n_2 are the order of LFG_1 and LFG_2, then from any $2(n_1 + n_2)$ consecutive bits of $[\mathbf{t}^{(1)}]$, it is possible by Berlekamp-Massey algorithm, to get the feedback polynomials of both LFSRs and so for LFGs. Now from Eq. (5), $\mathbf{s}_i = (\mathbf{t}_{n_2+i} - F_2(\mathbf{t}_i, \mathbf{t}_{i+1}, \ldots, \mathbf{t}_{n_2-1+i}))$ *mod* 2^m, for $i \geq 0$ and so the initial states $\mathbf{s}_0, \ldots, \mathbf{s}_{n_1-1}$ can be computed, once $[\mathbf{t}]$ is known.

Since $\mathbf{t}_0, \ldots, \mathbf{t}_{n_2-1}$ are assumed to be part of the secret key, they should not be used as keystream words. Let us assume the keystream words $\{\mathbf{t}_{n_2+i}\}_{i\geq 0}$ are known. Then using the Berlekamp-Massey algorithm on $[\mathbf{t}^{(1)}]$, it is again possible to find the feedback polynomial of both the LFGs. Now $\mathbf{s}_{n_2+i}$ can be computed as $\mathbf{s}_{n_2+i} = (\mathbf{t}_{2n_2+i} - F_2(\mathbf{t}_{n_2+i}, \ldots, \mathbf{t}_{2n_2-1+i}))$ *mod* 2^m, for $i \geq 0$. Again $\mathbf{s}_{n_2+n_1-1} = F_1(\mathbf{s}_{n_2-1}, \mathbf{s}_{n_2}, \ldots, \mathbf{s}_{n_2+n_1-2})$ and $F_1(.)$ is linear, thus $\mathbf{s}_{n_2-1}$ can be calculated from it. Using the same procedure, it is possible to retrieve all initial states of LFG_1. Here also same $2(n_1 + n_2)$ consecutive words of $[\mathbf{t}]$ is sufficient to mount this attack. This attack is summarized as follows.

Algorithm 1 *Attack on LFG-based cascade systems*

Input: The output word sequence $[\mathbf{t}]$

Output: Recover the initial states of LFGs

1 **if** $[\boldsymbol{t}] \neq 0$ **then**

2 *Collect* $[\boldsymbol{t}^{(1)}]$*, the first LSB of each word of* $[\boldsymbol{t}]$ *Apply the Berlekamp-Massy algorithm on* $[\boldsymbol{t}]$ *and get the feedback polynomial of LFGs.*

3 *Using Eq. (5), calculate initial states of LFGs*

4 **else**

5 *All states of LFGs are* ***0***

This attack is applicable as long as the bits of $[\mathbf{t}^{(1)}]$ satisfy the linear recurring relation of order $(n_1 + n_2)$ or less. One easy solution is to use a nonlinear bijective function $g(.)$ to destroy the linear structure of $[\mathbf{t}^{(1)}]$ such as the use of an S box. Thus, this cryptanalytic attack is not applicable to CS_3. In the case of an LFG-based cascade system, simply rotation by a nonzero value will also destroy linear relation in the 1st LSB of $[\mathbf{t}]$.

6 Conclusion

WFSRs are useful as pseudorandom sequence generation. In this paper, we have given a general expression for WFSRs. In Table 4, some well-known FSRs are shown as the special case of WFSR by putting different restrictions. Column 1 of Table 4 tells the name of the FSR. Columns 2 and 3 give the value of the degree of feedback function F and the word size m, respectively. Column 4 tells which operation is used for $\odot$ in Eq. (5) and column 5 says, the multiplication operation used when $deg(F) > 1$.

FSR	$deg(F)$	m	$\odot$	$\prod$
LFSR	1	1	$\oplus$	
NLFSR	> 1	1	$\oplus$	&
LFG	1	> 1	$+$	
MRMM	1	> 1	$\oplus$	

Table 4 Special cases of different WFSRs

Appendix A

S[256] = [0x9A, 0x85, 0xAF, 0xBC, 0x00, 0xAB, 0x89, 0xC1, 0x6D, 0x7F, 0xB8, 0x1C, 0x13, 0x30, 0x37, 0xA6, 0xDB, 0x71, 0xD2, 0x54, 0x31, 0x32, 0xD9, 0xFC, 0xE4, 0x99, 0xCF, 0x15, 0xF6, 0x34, 0x84, 0xBF, 0x3A, 0xAA, 0x6F, 0xB3, 0xBE, 0xEE, 0xFD, 0xD3, 0x4F, 0x23, 0xCE, 0x9B, 0x3B, 0xCC, 0xA9, 0x04, 0xA5, 0xF2, 0x1B, 0xC3, 0x8A, 0xF3, 0x5D, 0x16, 0xAC, 0x47, 0x77, 0x11, 0x2F, 0x1E, 0x08, 0x2E, 0xCA, 0x09, 0x38, 0x40, 0xDA, 0xE7, 0xB4, 0xE6, 0x88, 0xED, 0xA0, 0xD1, 0x29, 0xE3, 0x3E, 0xC5, 0x70, 0x58, 0x46, 0x91, 0x0A, 0xAD, 0x1A, 0xA1, 0x4A, 0xCD, 0x0B, 0x9F, 0xB9, 0x20, 0xD5, 0x42, 0x05, 0xF1, 0x14, 0x75, 0xE0, 0xAE, 0x36, 0xC6, 0x92, 0x8E, 0x94, 0x26, 0x79, 0xB5, 0xDC, 0xB6, 0x81, 0x3C, 0x74, 0xC4, 0xD6, 0x19, 0xE5, 0xA8, 0xFA, 0x12, 0xD8, 0xDE, 0x69, 0x49, 0x7A, 0x44, 0xB2, 0xD0, 0xE9, 0xF0, 0xCB, 0x56, 0x4D, 0x07, 0xBA, 0x97, 0x24, 0xE2, 0x8D, 0x59, 0xA4, 0xA3, 0x93, 0x86, 0x21, 0xF8, 0x3D, 0x83, 0x3F, 0x28, 0x43, 0xA7, 0x9C, 0x98, 0x72, 0x7D, 0x8F, 0xC8, 0xEF, 0x2B, 0x41, 0x90, 0xF4, 0xEC, 0x1F, 0xF9, 0x68, 0x51, 0x01, 0x0C, 0x60, 0x62, 0xBD, 0x5A, 0x48, 0x52, 0x8B, 0x67, 0xE8, 0xFE, 0xA2, 0x53, 0xB1, 0xC2, 0xC7, 0x96, 0x87, 0x22, 0x4C, 0x45, 0x55, 0x4B, 0x95, 0xEB, 0xDD, 0xFF, 0xD7, 0x5E, 0x1D, 0x9D, 0x80, 0x6A, 0x76, 0xD4, 0x0E, 0x4E, 0x9E, 0x50, 0x2A, 0x82, 0x27, 0x7C, 0x7E, 0x61, 0x6B, 0x6E, 0x0D, 0xB7, 0x03, 0x5C, 0x8C, 0xFB, 0x17, 0x2D, 0x73, 0xC0, 0x57, 0x18, 0x0F, 0xDF, 0x06, 0x33, 0xE1, 0x7B, 0x25, 0x66, 0x39, 0x78, 0x10, 0x6C, 0x64, 0xF7, 0xBB, 0xB0, 0x02, 0x35, 0x63, 0x5F, 0xC9, 0x2C, 0xEA, 0x65, 0xF5, 0x5B]

References

[1] Bassham, L., Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, N., Dray, J.: A statistical test suite for random and pseudo random number generators for cryptographic applications, Special Publication (NIST SP) - 800-22 Rev 1a, 2010, [Online], available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf

[2] Bishoi, S.K., Haran, H.K., Hasan, S.U.: A note on the multiple-recursive matrix method for generating pseudorandom vectors, Discrete Applied Mathematics, vol. 222, pp. 67-75 (2017)

[3] Bishoi, S.K., Senapati, K., Shankar, B.R.: Shrinking generators based on

σ-LFSRs, Discrete Applied Mathematics, vol. 285, pp. 493-500 (2020)

[4] Bishoi, S.K., Senapati, K., Shankar, B.R.: *Bitstream generators using Multiple-Recursive Matrix Methods*, Sept 2022, doi: https://doi.org/10.21203/rs.3.rs-2105578/v1

[5] Brent, R. P.: On The Periods of Generalized Fibonacci Recurrences, Mathematics of Computation, Vol. 63, Number 207, July 1994, pp. 389-401

[6] CAESAR Competition for Authenticated Encryption: Security, Applicability, and Robustness (2012), http://Competitions.cr.yp.to/caesar.html

[7] Cannière, C., Preneel, B.: Trivium, in New Stream Cipher Designs: The eSTREAM Finalists (Lecture Notes in Computer Science), vol. 4986, Berlin, Germany: Springer-Verlag, pp. 244–266 (2008)

[8] Canteaut, A.: Open problems related to algebraic attacks on stream ciphers, in Proc. Workshop Coding Theory Cryptograph. (Lecture Notes in Computer Science), vol. 3969, Berlin, Germany: Springer-Verlag, pp. 120-134 (2006)

[9] Chetry, M.K., Bishoi, S.K., Matyas, V.: When lagged fibonacci generators jump, Discrete Applied Mathematics, vol. 267, pp. 64-72 (2019)

[10] Courtois, N.: Fast algebraic attacks on stream ciphers with linear feedback, in Advances in Cryptology-CRYPTO (Lecture Notes in Computer Science), vol. 2729, Berlin, Germany: Springer-Verlag, pp. 176-194 (2003)

[11] Courtois, N., Meier, W.: Algebraic attacks on stream ciphers with linear feedback, in Advances in Cryptology-EUROCRYPT (Lecture Notes in Computer Science), vol. 2656, Berlin, Germany: Springer-Verlag, pp. 345-359 (2003)

[12] Cryptographic competitions eSTREAM: the ECRYPT Stream Cipher Project, http://Competitions.cr.yp.to/estream.html

[13] Golomb, S.W.: Shift Register Sequences, Revised Edition, Aegean Park Press, Laguna Hills (1982)

[14] Hamann, M., Krause, M., Meier, W.: Lizard: a lightweight stream cipher for power-constrained devices. IACR Trans Symmetric Cryptol, pp. 45-79 (2017)

[15] Hasan, S.U., Panario, D., Wang, Q.: Nonlinear vectorial primitive recursive sequences, Cryptography and Communications, 10, pp. 1075-1090 (2018)

[16] Hell, M., Johansson, T., Maximov, A., Meier, W.: The Grain family of

stream ciphers, in New Stream Cipher Designs: The eSTREAM Finalists (Lecture Notes in Computer Science), vol. 4986, Berlin, Germany: Springer-Verlag, pp. 179–190 (2008)

[17] Hu, H: ACORN: A Lightweight Authenticated Cipher (v3), Submission to the CAESAR competition: https://competitions.cr.yp.to/round3/acornv3.pdf (2016)

[18] Lidl, R., Niederreiter, H.: Finite Fields, Encyclopedia of Mathematics and its Applications 20, Cambridge University Press, Cambridge, England (1997)

[19] Marsaglia, G., Tsay, L.: Matrices and the structure of Random Number Sequences. Linear Algebra and its Applications 67: 147-156, 1985

[20] Menezes, A.J., van Oorschot, P. C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press (1997)

[21] Mikhalev, V., Armknecht, F., Muller, C.: On ciphers that continuously access the non-volatile key, IACR Trans Symmetric Cryptol, pp. 52–79 (2016)

[22] Niederreiter, H.: The multiple-recursive matrix method for pseudorandom number generation, Finite Fields Appl. 1, pp. 3-30 (1995)

[23] Preneel, B.: NESSIE Project. In: van Tilborg, H.C.A. (eds) Encyclopedia of Cryptography and Security, Springer, Boston, MA. https://doi.org/10.1007/0-387-23483-7_271

[24] Rueppel, R.A., Staffelbach, O.J.: Products of linear recurring sequences with maximum complexity, IEEE Transactions on Information Theory, 33, pp. 124-131 (1987)

[25] Stinson, D.R.: Cryptography Theory and Practice. Chapman & Hall/CRC, Third Edition (2006)

[26] Tsaban, B., Vishne, U.: Efficient feedback shift registers with maximal period, Finite Fields Appl. 8, pp. 256-267 (2002)

[27] Zeng, G., Han, W., He, K.: Word-oriented feedback shift register: σ-LFSR (2007)