On Analysis and Synthesis of (n,k)-Non-Linear Feedback Shift Registers

Elena Dubrova   Maxim Teslenko   Hannu Tenhunen
Royal Institute of Technology (KTH),
Electrum 229, 164 46 Kista, Sweden
{elena, maximt, hannu}@imit.kth.se

Abstract— **Non-Linear Feedback Shift Registers (NLFSRs) have been proposed as an alternative to Linear Feedback Shift Registers (LFSRs) for generating pseudo-random sequences for stream ciphers. In this paper, we introduce (n,k)-NLFSRs which can be considered a generalization of the Galois type of LFSR. In an (n,k)-NLFSR, the feedback can be taken from any of the n bits, and the next state functions can be any Boolean function of up to k variables. Our motivation for considering this type NLFSRs is that their Galois configuration makes it possible to compute each next state function in parallel, thus increasing the speed of output sequence generation. Thus, for stream cipher application where the encryption speed is important, (n,k)-NLFSRs may be a better alternative than the traditional Fibonacci ones. We derive a number of properties of (n,k)-NLFSRs. First, we demonstrate that they are capable of generating output sequences with good statistical properties which cannot be generated by the Fibonacci type of NLFSRs. Second, we show that the period of the output sequence of an (n,k)-NLFSR is not necessarily equal to the length of the largest cycle of its states. Third, we compute the period of an (n,k)-NLFSR constructed from several parallel NLFSRs whose outputs are XOR-ed and show how to maximize this period. We also present an algorithm for estimating the length of cycles of states of (n,k)-NLFSRs which uses Binary Decision Diagrams for representing the set of states and the transition relation on this set.**

I. Introduction

Information security is of paramount importance to many institutions of our society: governments, military, financial, businesses, etc. Many confidential information about research, products, financial status, customers, or employees, is nowadays processed and stored on computers, or transmitted to other computers.

In order to protect the confidential information from unauthorized or accidental discloser, cryptographic methods are applied. A common approach is to use a symmetric stream cipher which combines plain text bits with a pseudo-random bit key-stream, typically by an XOR operation. Encrypted information can be transformed back into its original form only by an authorized user possessing the cryptographic key.

The pseudo-random bit sequences are often generated using *Linear Feedback Shift Registers (LFSRs)*. Advantages of LFSRs include the ease of implementation, simplicity, speed, and the ability to generate a maximal cycle sequence with the same uniform statistical distribution of 0's and 1's as in a truly random sequence [1]. The main disadvantage of LFSRs is their linearity, leading to a relatively easy cryptanalysis [2].

A common solution to this problem in LFSR-based stream ciphers is to feed the outputs of several parallel LFSRs into a non-linear Boolean function to form a combination generator [3], [5]. The combining function has to be carefully selected to ensure the security of the resulting scheme, for example, in order to prevent correlation attacks [4]. Other approaches are to combine several bits from the LFSR state using a non-linear function, or to use the irregular clocking of the LFSR [6], [7]. Important LFSR-based stream ciphers include A5/1 stream cipher which used to provide over-the-air communication privacy in the GSM cellular telephone standard [8], E0 stream cipher which is used in the Bluetooth protocol [9], [10], and the shrinking generator [11].

As another alternative, *Non-Linear Feedback Shift Register (NLFSR)* whose current state is a non-linear function of its previous state can be used. A number of different implementations of NLFSR-based stream ciphers for RFID and smartcards applications have been proposed, including Achterbahn [12], Grain [13], [14], KeeLoq [15], Trivium [16], VEST [17], and [18]. NLFSRs have been shown to be more resistant to cryptanalytic attacks than LFSRs [19], [20]. However, construction of large NLFSRs with guaranteed long periods remains an open problem. A systematic algorithm for NLFSR synthesis has not been discovered so far. Only solutions to some special cases have been presented [1], [21], [22], [23], [24], [25], [26], [27].

Most commonly, the *Fibonacci* implementation of NLFSR, shown in Figure 1, is used. The Fibonacci type of NLFSR consists of a number of bits numbered from right to left as $0,\ldots,n-1$ with feedback from each bit to the $n-1$th bit. At each clocking instance, the value of the bit i is moved to the bit $i-1$. The value of the bit 0 becomes the output of the register. The new value of the bit $n-1$ is computed as some non-linear function of the previous values of bits $0,\ldots,n-1$, depending on the feedback function used.

In this paper, we introduce an alternative type of NLFSRs, which we call (n,k)-*NLFSRs*. An (n,k)-NLFSR can be considered as a generalization of the Galois type of LFSR to the non-linear case. Each bit i in an (n,k)-NLFSR is updated according to its next-state function, which is a non-linear function of the bit $i+1$ and up to k other bits. Thus, in contrast to the Fibonacci NLFSR in which feedback is applied to the $n-1$th bit only, in (n,k)-NLFSRs feedback is potentially applied to every bit. For the case of $k=n$, an (n,n)-NLFSR can be considered as a special case of a general autonomous n-state machine [1]. We are not aware of any work on (n,k)-NLFSRs for the case of $k<n$.

Our motivation for considering (n,k)-NLFSRs is that the Galois type of their configuration gives us a potential opportunity to increase the speed of output sequence generation. In (n,k)-NLFSRs, next state functions of individual bits are placed within the register, making possible to compute each next state function in parallel. Therefore, the propagation time is reduced to that of smaller functions of individual bits rather than the time of a large feedback function of the Fibonacci type of NLFSRs. We target stream ciphers application in which high encryption speed is very important.

We derive a number of properties of (n,k)-NLFSRs. First, we show that (n,k)-NLFSRs are capable of generating output sequences with good statistical properties which cannot be generated by the Fibonacci type of NLFSRs. While every Fibonacci NLFSR has a matching (n,k)-NLFSRs, the opposite is not true.

Second, we show that the period of the output sequence of an (n,k)-NLFSR is not necessarily equal to the length of the longest cyclic

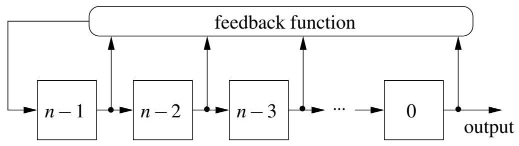

Fig. 1. An n-bit Fibonacci NLFSR.

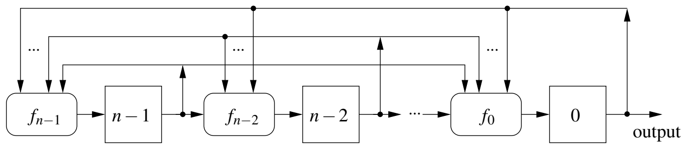

Fig. 2. An (n,k)-NLFSR. Each function f_i, $i \in \{0,1,\dots,n-1\}$ has up to k inputs.

sequence of its consecutive states. These two notions are always the same for the Fibonacci type of NLFSRs.

Similarly to LFSRs and the Fibonacci type of NLFSRs, one can construct a larger (n,k)-NLFSR from several smaller ones working in parallel by XOR-ing their outputs. We show how to select these NLFSRs in order to maximize the period.

Another contribution of this paper is an algorithm for estimating the length of the cycles of states of (n,k)-NLFSRs. This algorithm uses reduced ordered Binary Decision Diagrams [28] for representing the set of states of an (n,k)-NLFSR and the transition relation on this set. Unlike other compressed representations of relations or sets, Binary Decision Diagrams perform the actual operations directly on the compressed representation, without decompressing its first. The efficiency of the presented algorithm makes it useful, for example, for analyzing a large number of randomly generated (n,k)-NLFSRs.

The paper is organized as follows. Section II describes main notions and definitions used in the sequel. Section III introduces (n,k)-NLFSRs. Section IV analyzes a relation between (n,k)-NLFSRs and the Fibonacci type of NLFSRs. In Section V, we investigate how the length of the longest cycle of states of an (n,k)-NLFSR is related to its period. In Section VI, we estimate the period of an (n,k)-NLFSR constructed from several NLFSRs working in parallel. Section VII presents an algorithm for computing the length of the cycles of states of (n,k)-NLFSRs and Section VIII evaluates it. Section IX concludes the paper and discusses open problems.

II. Background

A. Properties of a good pseudo-random sequence

In order to look like a random sequence, a pseudo-random sequence should satisfy the following properties, called Golomb's postulates [1]:

G1. The number of zeros and ones should be as equal as possible per period.

G2. Half of the runs in a period have length 1, one-quarter have length 2, ... , $1/2^i$ have length i. Furthermore, for any length, half of the runs are blocks and the other half are gaps. A *block* is a subsequence of 1's and a *gap* is a subsequence of 0'; either type of subsequences is a *run*.

G3. The out-of-phase autocorrelation $AC(k)$ has the same value for all k. The *autocorrelation* is defined as

$$AC(k) = \frac{\text{(Agreements - Disagreements)}}{p},$$

where a sequence of period p is compared to its shift by k bits. The autocorrelation is *out-of-phase* if p does not divide k evenly.

To be of practical use for cryptography, a good pseudo-random sequence is also required to satisfy the following properties [29]:

1) The period should be very long, $\approx 10^{50}$ as minimum.
2) The sequence should be easy to generate, for fast encryption.
3) The cryptosystem based on the sequence should be cryptographically secure against chosen plain text attack.

B. Transition relation

A *transition relation* defines the next state values of a transition system in terms of the current state values. We derive the transition relation in the standard way, by making two copies of the set of state variables: $s = (x_0, x_1, \dots, x_{n-1})$, denoting the variables of the current state, and $s^+ = (x_0^+, x_1^+, \dots, x_{n-1}^+)$, denoting the variables of the next state [30]. For example, the transition relation of the Fibonacci NLFSR in Figure 1 is given by the following characteristic formula:

$$T(s,s^+) = (x_{n-1}^+ \leftrightarrow f(x_0,x_1,\dots,x_{n-1})) \wedge (\bigwedge_{i=0}^{n-2} (x_i^+ \leftrightarrow x_{i+1}^+)).$$

C. Reachability analysis

In *forward reachability* analysis [30], a sequence of formulas $F_i(s)$ representing the set of states that can be reached from a given set of initial states *Init* in i steps is computed as:

$$F_0 = Init,$$
$$F_{i+1}(s^+) = \exists s.(T(s,s^+) \wedge F_i(s)).$$

The sequence generation is terminated when the fixed point is reached for some p:

$$\bigvee_{i=1}^{p} F_i(s) \rightarrow \bigvee_{i=1}^{p-1} F_i(s).$$

In *backward reachability* analysis [30], a sequence of formulas $B_i(s)$ representing the set of states from which a given set of final states *Final* can be reached in i steps is computed as:

$$B_0 = Final,$$
$$B_{i+1}(s) = \exists s^+.(T(s,s^+) \wedge B_i(s^+)).$$

The sequence generation is terminated when the fixed point is reached for some p:

$$\bigvee_{i=1}^{p} B_i(s) \rightarrow \bigvee_{i=1}^{p-1} B_i(s).$$

D. Binary Decision Diagrams

A *Binary Decision Diagram (BDD)* is a rooted directed acyclic graph which consists of decision nodes and two terminal nodes called 0- and 1-terminal [28]. Each decision node is labeled by a Boolean variable and has two children called *low* and *high* child. The edge from a node to a low (high) child represents an assignment of the variable to 0 (1). A path from the root node to the 1 (0)-terminal node represents an assignment of variables for which the represented Boolean function evaluates to 1 (0).

A BDD is *ordered* if different variables appear in the same order on all paths from the root to the terminal nodes. A BDD is *reduced* if all isomorphic subgraphs are merged, and any node whose two children are isomorphic is eliminated. The advantage of a reduced ordered BDD is that, for a chosen order of variables, it is unique for the represented function. This property makes reduced ordered BDDs

TABLE I
$(4,k)$-NLFSRs WITH THE PERIOD 15.

N	Next state functions				Output sequence
	f_3	f_2	f_1	f_0	
1	x_0	$x_3 \oplus 1 \oplus x_0 \oplus x_1 \oplus x_0x_1$	$x_2 \oplus 1 \oplus x_3 \oplus x_1$	$x_1 \oplus 1 \oplus x_2 \oplus x_0 \oplus x_2x_0$	001100011110101
2	x_0	$x_3 \oplus x_0x_2$	$x_2 \oplus x_3 \oplus x_1$	$x_1 \oplus x_0 \oplus x_0x_2$	110011110001010
3	x_0	$x_3 \oplus x_0x_2$	$x_2 \oplus 1 \oplus x_3 \oplus x_0x_3$	$x_1 \oplus 1 \oplus x_2 \oplus x_0 \oplus x_2x_0$	110100100011110
4	x_0	$x_3 \oplus x_0x_2$	$x_2 \oplus 1 \oplus x_0 \oplus x_1x_0$	$x_1 \oplus 1 \oplus x_2 \oplus x_0x_2$	110100010011110
5	x_0	$x_3 \oplus 1 \oplus x_1 \oplus x_2 \oplus x_1x_2$	$x_2 \oplus x_1 \oplus x_0x_1$	$x_1 \oplus 1 \oplus x_0 \oplus x_0x_2$	010001111001101
6	x_0	$x_3 \oplus x_1x_2$	$x_2 \oplus x_1 \oplus x_1x_0$	$x_1 \oplus x_2x_0$	100011001111010
7	x_0	$x_3 \oplus x_1x_2$	$x_2 \oplus 1 \oplus x_0$	$x_1 \oplus 1 \oplus x_2 \oplus x_3 \oplus x_2x_3$	110001001011110
8	x_0	$x_3 \oplus 1 \oplus x_1 \oplus x_2 \oplus x_1x_2$	x_2	$x_1 \oplus 1 \oplus x_2 \oplus x_0 \oplus x_2x_0$	010111100011001
9	x_0	$x_3 \oplus x_1 \oplus x_0x_1$	$x_2 \oplus 1 \oplus x_3 \oplus x_1$	$x_1 \oplus x_2 \oplus x_0x_2$	011011110001001
10	x_0	$x_3 \oplus x_2 \oplus x_0x_2$	$x_2 \oplus 1 \oplus x_0 \oplus x_1 \oplus x_0x_1$	$x_1 \oplus 1 \oplus x_0 \oplus x_0x_2$	010011110001101
11	x_0	$x_3 \oplus x_2 \oplus x_1x_2$	$x_2 \oplus x_3x_1$	$x_1 \oplus x_2 \oplus x_3 \oplus x_2x_3$	011011110010001
12	x_0	$x_3 \oplus x_2 \oplus x_1x_2$	$x_2 \oplus x_0x_1$	$x_1 \oplus x_2 \oplus x_0x_2$	001101111010001
13	x_0	$x_3 \oplus x_2 \oplus x_1x_2$	$x_2 \oplus x_0$	$x_1 \oplus x_2 \oplus x_0x_2$	010111100110001
14	x_0	$x_3 \oplus x_2 \oplus x_2x_0$	x_2	$x_1 \oplus x_2x_0$	110010001011110
15	x_0	$x_3 \oplus 1 \oplus x_1 \oplus x_2 \oplus x_1x_2$	$x_2 \oplus 1 \oplus x_0 \oplus x_1x_0$	$x_1 \oplus 1 \oplus x_0 \oplus x_2 \oplus x_2x_0$	100100011011110
16	$x_0 \oplus 1 \oplus x_1 \oplus x_2 \oplus x_1x_2$	x_3	x_2	$x_1 \oplus x_2 \oplus x_3$	001010011110001
17	$x_0 \oplus 1 \oplus x_1 \oplus x_2 \oplus x_1x_2$	x_3	$x_2 \oplus x_3 \oplus x_1x_3$	$x_1 \oplus x_3$	001001111010001
18	$x_0 \oplus x_1x_2$	$x_3 \oplus x_0x_2$	x_2	$x_1 \oplus 1 \oplus x_3 \oplus x_0$	100100111100010
19	$x_0 \oplus 1 \oplus x_1 \oplus x_2 \oplus x_1x_2$	$x_3 \oplus 1 \oplus x_0 \oplus x_2 \oplus x_0x_2$	$x_2 \oplus x_0x_1$	$x_1 \oplus x_3 \oplus x_0$	010100111100100
20	$x_0 \oplus x_1x_2$	$x_3 \oplus x_0x_2$	x_2	$x_1 \oplus 1 \oplus x_0 \oplus x_2 \oplus x_2x_0$	101100011110010
21	$x_0 \oplus 1 \oplus x_1 \oplus x_2 \oplus x_1x_2$	$x_3 \oplus 1 \oplus x_1 \oplus x_2 \oplus x_1x_2$	x_2	$x_1 \oplus x_2$	011001011110001
22	$x_0 \oplus x_1x_2$	$x_3 \oplus 1 \oplus x_1 \oplus x_2 \oplus x_1x_2$	$x_2 \oplus 1 \oplus x_1$	$x_1 \oplus 1 \oplus x_3 \oplus x_0$	100100011110010
23	$x_0 \oplus x_1x_2$	$x_3 \oplus 1 \oplus x_1 \oplus x_2 \oplus x_1x_2$	x_2	$x_1 \oplus 1 \oplus x_2$	100011110100110
24	$x_0 \oplus x_1x_2$	$x_3 \oplus 1 \oplus x_0$	$x_2 \oplus x_3 \oplus x_3x_1$	$x_1 \oplus 1 \oplus x_3 \oplus x_0$	100101000111100
25	$x_0 \oplus 1 \oplus x_1 \oplus x_2 \oplus x_1x_2$	$x_3 \oplus x_2$	$x_2 \oplus x_0x_1$	$x_1 \oplus 1 \oplus x_2 \oplus x_3$	100111100101000
26	$x_0 \oplus 1 \oplus x_1 \oplus x_3 \oplus x_1x_3$	x_3	x_2	$x_1 \oplus x_2 \oplus x_3$	001111010010001
27	$x_0 \oplus 1 \oplus x_1 \oplus x_3 \oplus x_1x_3$	x_3	x_2	$x_1 \oplus 1 \oplus x_2 \oplus x_3$	101111001000100
28	$x_0 \oplus 1 \oplus x_1 \oplus x_3 \oplus x_1x_3$	x_3	$x_2 \oplus x_3 \oplus x_1x_3$	$x_1 \oplus x_2 \oplus x_3 \oplus x_2x_3$	001001011110001
29	$x_0 \oplus 1 \oplus x_1 \oplus x_3 \oplus x_1x_3$	x_3	$x_2 \oplus x_3x_1$	$x_1 \oplus 1 \oplus x_2 \oplus x_3 \oplus x_2x_3$	100100011110100
30	$x_0 \oplus 1 \oplus x_1 \oplus x_3 \oplus x_1x_3$	$x_3 \oplus x_0x_1$	$x_2 \oplus 1 \oplus x_0$	$x_1 \oplus 1 \oplus x_2 \oplus x_0$	011000101111001
31	$x_0 \oplus 1 \oplus x_2 \oplus x_3 \oplus x_2x_3$	x_3	$x_2 \oplus x_1 \oplus x_3x_1$	$x_1 \oplus x_3$	001000101111001
32	$x_0 \oplus x_2x_3$	$x_3 \oplus x_1 \oplus x_2 \oplus x_1x_2$	$x_2 \oplus x_1 \oplus x_1x_3$	$x_1 \oplus x_0$	100011010011110

particularly useful in formal verification, since the equivalence of two BDDs can be checked in constant time. Other logical operations, such as conjunction, disjunction, negation, existential quantification, universal quantification, can be performed on BDDs in linear or quadratic time in the size of the graphs [31].

III. (n,k)-NLFSR DEFINITION

An (n,k)-NLFSR can be considered a generalization of the Galois type of LFSRs to the non-linear case. The general structure is shown in Figure 2. In the Galois type of LFSRs, the feedback is taken from the 0th bit only and the next state functions f_i are restricted to the 2-input XORs. In an (n,k)-NLFSR, the feedback can be taken from any bit. The next state function can be any Boolean function of up to k variables.

More formally, an (n,k)-NLFSR can be defined as follows. Let x_i and x_i^+ be state variables representing the current and the next states of the bit i, respectively, and let $f_i : \{0,1\}^k \rightarrow \{0,1\}$, $1 \leq k \leq n$, be the next state function of the bit i. The function f_i always depends on the bit $(i+1) mod\ n$. It also depends on up to $k-1$ other bits. If the indexes of these bits are $i_1, \ldots, i_{k-1}$, $i_j \in \{0,1,\ldots,n-1\}$, $j \in \{1,2,\ldots,k-1\}$, then the next state value of the ith bit is given by:

$$x_i^+ = f_i(x_{(i+1) mod\ n}, x_{i_1}, \ldots, x_{i_{k-1}}).$$

The output of the (n,k)-NLFSR is the output of its 0th bit.

The *state* of an (n,k)-NLFSR is defined by the ordered set of values of its state variables $(x_0, x_1, \ldots, x_{n-1})$. Since an (n,k)-NLFSR is deterministic and finite, any sequence of consecutive states eventually converges to either a single state, or a cycle of states.

The *period* of an (n,k)-NLFSR is the length of the longest cyclic output sequence it produces. The period of an (n,k)-NLFSR can be less of equal to 2^n [32].

Throughout the paper, we use the algebraic normal form to represent Boolean functions. The *algebraic normal form (ANF)* of a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ is a polynomial in $GF(2)$ of type

$$f(x_0, \ldots, x_{n-1}) = \sum_{(i_0,\ldots,i_{n-1}) \in \{0,1\}^n} c(i_0, \ldots, i_{n-1}) \cdot x_0^{i_0} \ldots x_{n-1}^{i_{n-1}}$$

where $c(i_0, \ldots, i_{n-1}) \in \{0,1\}$.

TABLE II
4-BIT FIBONACCI NLFSRs WITH THE PERIOD 15.

N	Feedback function	Output sequence
1	$x_0 \oplus x_1$ (LFSR $x^4 + x + 1$)	111000100110101
2	$x_0 \oplus x_1 \oplus x_2 \oplus x_1x_2$	111000101101001
3	$x_0 \oplus x_2 \oplus x_3 \oplus x_1x_2$	111010001100101
4	$x_0 \oplus x_1 \oplus x_2 \oplus x_1x_3$	111011000101001
5	$x_0 \oplus x_2 \oplus x_1x_2 \oplus x_1x_3$	111010001011001
6	$x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_1x_2 \oplus x_1x_3$	111000110100101
7	$x_0 \oplus x_1 \oplus x_1x_2 \oplus x_2x_3$	111010110001001
8	$x_0 \oplus x_1 \oplus x_2 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3$	111000101001101

IV. RELATION BETWEEN (n,k)-NLFSRs AND THE FIBONACCI TYPE OF NLFSRs

If the next state functions f_i of all bits from 0 to $n-2$ are of type $f_i = x_{i+1}$ and $k = n$, then an (n,k)-NLFSR reduces to the Fibonacci type of NLFSRs shown in Figure 1.

Being a generalization of the Fibonacci type of NLFSRs, (n,k)-NLFSRs are capable of generating all output sequences which can be generated by the Fibonacci ones. However, (n,k)-NLFSRs can also produce many output sequences with good statistical properties which cannot be generated by the Fibonacci NLFSRs. To demonstrate this, in Table I we show 32 $(4,3)$-NLFSRs generating output sequences with the period 15 which cannot be produced by any of 4-bit Fibonacci NLFSRs. All output sequences in Table II satisfy the 1st and 2nd postulates of Golomb. For a reference, Table II lists all possible output sequences with the period 15 which can be generated by the Fibonacci NLFSRs, excluding reverse (i.e. mirror image) and complemented cases.

In the Fibonacci NLFSRs, any output sequence with the period 2^n-1 always satisfies the 1st and 2nd postulates of Golomb. This is not the case for (n,k)-NLFSRs. Some of the output sequences with the period 2^n-1 generated by (n,k)-NLFSRs do not satisfy the 1st or 2nd postulates. The 3rd postulate of Golomb is never satisfied by either type of NLFSRs because the sum of two shifted NLFSR sequences is not always another output sequence.

V. Relation between the length of the longest cycle of states of an (n,k)-NLFSR and its period

Another specific property of (n,k)-NLFSRs is that the period of their output sequence of is not necessarily equal to the length of the longest cycle of their states. For example, a $(4,k)$-NLFSR going through the following 16 states:

$$1,3,0,2,8,5,7,6,9,11,4,10,14,13,15,12$$

generates the output sequence 1100011011000110 which has the period 8. For the Fibonacci type of NLFSRs, the period is always equal to the length of the longest cycle of states.

It is easy to prove that this problem never occurs if the number of 0's and 1's in the output sequence of an (n,k)-NLFSRs differs by 1.

VI. Synthesis of (n,k)-NLFSRs by composition

Similarly to LFSRs and the Fibonacci type of NLFSRs, we can construct an (n,k)-NLFSR with a guaranteed long period by composing several smaller NLFSRs working in parallel and combining their outputs using an XOR. The length of the period of such an NLFSR can be easily derived from the result for general finite deterministic transition systems, e.g. [33], as follows.

Let $R_1,R_2,\ldots,R_m$ be $(n_1,k_1),(n_2,k_2),\ldots,(n_m,k_m)$-NLFSRs, respectively, and R be an (n,k)-NLFSR which they compose, where $n=n_1+n_2+\ldots+n_m$ and $k=max(k_1,k_2,\ldots,k_m)$. Let N_i denote the number of different cycles of states produced by R_i, and let L_{ij} denote the length of the jth cycle of states of R_i, $i=\{1,2,\ldots,m\}$, $j=\{1,2,\ldots,N_i\}$. The state of R is defined as a concatenation of states of $R_1,R_2,\ldots,R_m$.

Theorem 1: The length of the longest cycle of states of an (n,k)-NLFSR R which is composed of m parallel (n_i,k_i)-NLFSRs R_i, $i\in\{1,2,\ldots,m\}$, is given by

$$L_{max}=\max_{\forall(i_1,\ldots,i_m)\in I}((L_{1i_1}\star L_{2i_2})\star L_{3i_3})\ldots\star L_{mi_m}$$

and the total number of cycles of states produced by R is given by

$$N=\sum_{\forall(i_1,\ldots,i_m)\in I}\prod_{j=2}^{m}(((L_{1i_1}\star L_{2i_2})\star L_{3i_3})\ldots\star L_{j-1i_{j-1}})\diamond L_{ji_j}$$

where "$\star$" is the least common multiple, "$\diamond$" is the greatest common divisor, and $I=I_1\times I_2\times\ldots\times I_m$ is the Cartesian product of sets $I_i=\{i_1,i_2,\ldots,i_{N_i}\}$, where the set I_i represents indexes of cycles of states of the NLFSR R_i.

We can see from the Theorem 1 that in order to maximize the length of the cycles of states produced by R we should choose the NLFSRs $R_1,R_2,\ldots,R_m$ so that their longest cycles of states $L_{1max},L_{2max},\ldots,L_{mmax}$ satisfy the condition:

$$gcd(L_{imax},L_{jmax})=1, \qquad (1)$$

for each pair $i,j\in\{1,2,\ldots,m\}$. In this case, the least common multiple of $L_{1max},L_{2max},\ldots,L_{mmax}$ equals to their product, since

$$lcm(a,b)=\frac{a*b}{gcd(a,b)}$$

for any non-zero integers a and b. The condition 1 is satisfied if, for each pair $i,j\in\{1,2,\ldots,m\}$, L_{imax} and L_{jmax} are relatively prime. Then, the longest cycle of states produced by R is given by

$$L_{max}=L_{1max}\cdot L_{2max}\cdot\ldots\cdot L_{mmax}.$$

It can be shown that, if the outputs of NLFSRs $R_1,R_2,\ldots,R_m$ are combined using an XOR, and the period of each R_i is equal to L_{imax} for all $i\in\{1,2,\ldots,m\}$, then the period of R is equal to L_{max}.

VII. Description of the Algorithm

In order to be able to compute the length of cycles of states of (n,k)-NLFSRs without simulating them, we developed a simple algorithm presented in this section. First we describe its general structure, and then discuss each of its steps in details.

The presented algorithm iterates through the following steps. Let S be initially the set of all states of an (n,k)-NLFSR.

1) Compute the set $C_{all}\subseteq S$ consisting of all states of all cycles of states of the (n,k)-NLFSR.
2) Choose at random a state $s_{init}\in S$. Compute the state $s_{cycle}\in S$ reachable from s_{init} in 2^n steps.
3) Compute the set $A\subseteq S$ of all states from which s_{cycle} can be reached in up to 2^n steps.
4) Compute the intersection $C=A\cap C_{all}$. The resulting set consists of all states of the cycle containing s_{cycle}. Its size is equal to the length of the cycle.
5) Update S as $S:=S-A$. If S is empty, the algorithm terminates. Otherwise, the steps 2, 3 and 4 are repeated starting from a randomly chosen state in S to find the remaining cycles.

A. Step 1

The intuitive idea behind the approach for computing the set C_{all} is the following. Suppose that we choose some state s_{init} of S at random and move from this state 2^n steps forward. The state which we reach, s_{cycle}, always belongs to one of the cycles of states of the (n,k)-NLFSR. This is because an (n,k)-NLFSR is deterministic and finite, and therefore it cannot take more than 2^n steps to reach a cycle. Furthermore, a cycle cannot be missed by making more steps because, once entered, it is never escaped.

Next, suppose that instead of making a 2^n step transition starting from one state only, we start from all states of S simultaneously. It is easy to see that the resulting set of states C_{all} contains all states of all cycles of the (n,k)-NLFSR. To measure their length, it remains to distinguish between the different cycles.

The key to an efficient implementation the idea described above is our ability to compute the states reachable from any state in 2^n steps in n iterations. This can be done by applying the *iterative squaring* technique introduced in [34] on the transition relation of the (n,k)-NLFSR. We use reduced ordered BDDs for representing the set of states and the transition relation implicitly using a formula in propositional logic. This makes the implementation of iterative

TABLE III

(n,k)-NLFSRs WITH THE PERIOD 2^n-1. THE NEXT STATE FUNCTIONS f_i FOR ALL $i \in \{0,\ldots,n-4\}$ ARE OF TYPE $f_i = x_{i+1}$.

(n,k)	Next state functions		
	f_{n-1}	f_{n-2}	f_{n-3}
$(5,4)$	$f_4 = x_0 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3$	$f_3 = x_4 \oplus x_0 \oplus x_1$	$f_2 = x_3$
$(6,4)$	$f_5 = x_0 \oplus x_2 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3$	$f_4 = x_5 \oplus x_2 \oplus x_3$	$f_3 = x_4 \oplus x_3$
$(7,4)$	$f_6 = x_0 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3$	$f_5 = x_6 \oplus x_1$	$f_4 = x_5 \oplus x_1$
$(8,4)$	$f_7 = x_0 \oplus x_1 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3$	$f_6 = x_7 \oplus x_1 \oplus x_2$	$f_5 = x_6 \oplus x_3$
$(9,4)$	$f_8 = x_0 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3$	$f_7 = x_8 \oplus x_0 \oplus x_4$	$f_6 = x_7$
$(10,5)$	$f_9 = x_0 \oplus x_3 \oplus x_6 \oplus x_{12}x_{13}x_6$	$f_8 = x_9 \oplus x_1 \oplus x_7 \oplus x_0x_{10}$	$f_7 = x_8 \oplus x_2 \oplus x_3 \oplus x_7$
$(11,4)$	$f_{10} = x_0 \oplus x_1 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3$	$f_9 = x_{10} \oplus x_1 \oplus x_2 \oplus x_6$	$f_8 = x_9 \oplus x_2 \oplus x_3$
$(12,4)$	$f_{11} = x_0 \oplus x_1 \oplus x_3 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3$	$f_{10} = x_{11} \oplus x_3 \oplus x_5$	$f_9 = x_{10} \oplus x_7 \oplus x_9$
$(13,4)$	$f_{12} = x_0 \oplus x_1 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3$	$f_{11} = x_{12} \oplus x_4 \oplus x_6$	$f_{10} = x_{11} \oplus x_9$
$(14,4)$	$f_{13} = x_0 \oplus x_1 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3$	$f_{12} = x_{13} \oplus x_2 \oplus x_8$	$f_{13} = x_{12} \oplus x_8$
$(15,4)$	$f_{14} = x_0 \oplus x_1 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3$	$f_{13} = x_{14} \oplus x_5 \oplus x_8$	$f_{12} = x_{13} \oplus x_9$
$(16,7)$	$f_{15} = x_0 \oplus x_6 \oplus x_7x_9 \oplus x_3x_6 \oplus x_3x_4x_{12}$	$f_{14} = x_{15} \oplus x_7 \oplus x_{10}$	$f_{13} = x_{14} \oplus x_{10}$
$(17,4)$	$f_{16} = x_0 \oplus x_2 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3$	$f_{15} = x_{16} \oplus x_7 \oplus x_{11}$	$f_{14} = x_{15} \oplus x_{14}$
$(18,8)$	$f_{17} = x_0 \oplus x_7 \oplus x_4x_{14} \oplus x_3x_{16} \oplus x_1x_4x_{15}$	$f_{16} = x_{17} \oplus x_7 \oplus x_9 \oplus x_{12}$	$f_{15} = x_{16}$
$(19,6)$	$f_{18} = x_0 \oplus x_8 \oplus x_{10} \oplus x_4x_6 \oplus x_8x_{10}$	$f_{17} = x_{18} \oplus x_1 \oplus x_{10}x_{17} \oplus x_{11}x_{15}x_{17}$	$f_{16} = x_{17}$
$(20,5)$	$f_{19} = x_0 \oplus x_3 \oplus x_3x_5 \oplus x_3x_6x_{11}$	$f_{18} = x_{19} \oplus x_8 \oplus x_6x_{11} \oplus x_6x_{12}$	$f_{17} = x_{18} \oplus x_4 \oplus x_{13} \oplus x_{17}$
$(21,7)$	$f_{20} = x_0 \oplus x_{11} \oplus x_1x_6 \oplus x_{12}x_{14}x_{18}$	$f_{19} = x_{20} \oplus x_{12} \oplus x_4x_9 \oplus x_9x_{16}$	$f_{18} = x_{19} \oplus x_{18}$

squaring very efficient. Unlike other compressed representations of relations or sets, BDDs perform the actual operations directly on the compressed representation, without decompressing its first.

Let $T^j(s,s^+)$ denote the transition relation describing the set of next states that can be reached from any current state in exactly j steps. For example, for $j=2$, the transition relation $T^2(s,s^+)$ is computed as follows:

$$T^2(s,s^+) = \exists s^{++}.(T(s,s^{++}) \wedge T(s^{++},s^+)). \qquad (2)$$

By applying the squaring iteratively, we can obtain $T^{2^j}(s,s^+)$ in j steps for any j [34].

We terminate the iterative computation of j if it becomes equal to n, or if

$$T^{2^j}(s,s^+) \;\rightarrow\; T^{2^{j-1}}(s,s^+),$$

for some $j \in \{1,\ldots,n-1\}$, i.e. if we reached a fixed point.

Using the resulting transition relation, we can compute the set of states that can be reached from any state in 2^n steps as:

$$C_{all}(s^+) = \exists s. T^{2^n}(s,s^+).$$

$C_{all}(s^+)$ represents the set of states of *all* cycles of states of the (n,k)-NLFSR.

B. Step 2

At the second step of the algorithm, we select some state s_{init} of S at random and compute the state reachable from s_{init} in 2^n steps as:

$$s_{cycle} = \exists s.(T^{2^n}(s,s^+) \wedge s_{init}), \qquad (3)$$

where s_{init} is represented by the formula

$$s_{init} = \bigwedge_{i=0}^{n-1} (x_i \;\leftrightarrow\; init_i(x_i)),$$

with $init_i(x_i)$ being the initial value of the state variable x_i, $i \in \{0,1,\ldots,n-1\}$.

Note that the equation 3 always returns a single state.

C. Step 3

After the state s_{cycle} of a cycle is identified, we compute all states from which this cycle can be reached as follows.

First, a transition relation $T_{0\ldots 2^j}(s,s^+)$ which defines the set of all next states that can be reached from any current state in *up to* 2^j steps, $j \in \{0,\ldots,n\}$, is calculated as follows:

$$T_0(s,s^+) = \textstyle\bigwedge_{i=1}^{n}(x_i^+ \;\leftrightarrow\; x_i),$$
$$T_{0\ldots 1}(s,s^+) = T(s,s^+) \vee T_0(s,s^+),$$
$$T_{0\ldots 2^j}(s,s^+) = T^2_{0\ldots 2^{j-1}}(s,s^+),$$

where T_0 is the transition relation which assigns the next state of any state to be the state itself, and $T^2_{0\ldots 2^{j-1}}(s,s^+)$ is the square of $T_{0\ldots 2^{j-1}}(s,s^+)$ computed similarly to the one in the equation (2).

As previously, we terminate the iterative computation of $T_{0\ldots 2^j}(s,s^+)$ if j becomes equal to n, or if

$$T_{0\ldots 2^j}(s,s^+) \;\rightarrow\; T_{0\ldots 2^{j-1}}(s,s^+),$$

for some $j \in \{1,\ldots,n-1\}$.

Using the resulting transition relation, we compute the set of states from which the state s_{cycle} can be reached in up to 2^n steps as

$$A(s) = \exists s^+.(T_{0\ldots 2^n}(s,s^+) \wedge s_{cycle}).$$

D. Step 4

Clearly, by intersecting A and C_{all} we obtain all states belonging to the cycle C. If $C = C_{all}$, then the (n,k)-NLFSR has a cycle of states of length $|C|$ and the algorithm terminates. Otherwise, we compute the difference $S - A$ and repeat the steps 2, 3 and 4 starting from a randomly selected state in the resulting set until S is exhausted.

VIII. Evaluation of the Algorithm

We have investigated whether the presented algorithms can be of assistance in finding (n,k)-NLFSRs with the period 2^n-1 by evaluating a large number of randomly generated NLFSRs. To bound the random search, we have imposed the following restrictions:

1) The next state functions f_i for all $i \in \{0,\ldots,n-4\}$ are of type $f_i = x_{i+1}$.
2) f_{n-1} is the XOR of the product x_0, up to two other single-variable products generated randomly, and the non-linear part $x_1x_2 \oplus x_1x_3 \oplus x_2x_3$. If no (n,k)-NLFSR with this structure and the period 2^n-1 can be found, then the non-linear part of f_{n-1} is of type $x_{i_1}x_{i_2} \oplus x_{i_3}x_{i_4} \oplus x_{i_5}x_{i_6}x_{i_7}$ where the indexes i_j are generated randomly.

3) f_{n-2} is a linear function consisting of the XOR of the product x_{n-1} and up to three other single-variable products generated randomly. If no NLFSR with with this structure and the period $2^n - 1$ can be found, then f_{n-2} also has a non-linear part of type $x_{i_1}x_{i_2} \oplus x_{i_3}x_{i_4} \oplus x_{i_5}x_{i_6}x_{i_7}$ where the indexes i_j are generated randomly.

4) f_{n-3} is constricted similarly to f_{n-2}.

Note, that the configuration of $x_{i_1}x_{i_2} \oplus x_{i_3}x_{i_4} \oplus x_{i_5}x_{i_6}x_{i_7}$ may get reduced if some indexes are equal. For example, if $i_1 = i_2$, then the product $x_{i_1}x_{i_2}$ reduces to a single-variable product.

Table III lists (n,k)-NLFSRs with the period $2^n - 1$ for the smallest value of k which we were able to find. Output sequences of all these NLFSRs satisfy the 1st and 2nd postulates of Golomb.

Current version of our algorithm is too memory consuming for functions larger than 21 variables. We are investigating a possibility to reduce the memory consumption by combining BDDs with Boolean circuits as in verification algorithms [35], [36].

IX. Conclusion

In this paper, we introduce (n,k)-NLFSRs and derive some of their properties. This work is a first step towards developing a general theory of (n,k)-NLFSRs. Many important open problem remain, including deriving a lower bound on k as a function of n and the period length, finding an algorithm for constructing (n,k)-NLFSRs with a guaranteed long period and the minimum k, validating the potential advantage of a higher encryption speed offered by (n,k)-NLFSRs, estimating security of (n,k)-NLFSRs and performing statistical tests to evaluate their output sequences.

References

[1] S. Golomb, *Shift Register Sequences*. Aegean Park Press, 1982.

[2] B. Schneier, "A self-study course in block-cipher cryptanalysis," *Cryptologia*, vol. XXIV, no. 1, pp. 18–33, 2000.

[3] M. Robshaw, "Stream ciphers," Tech. Rep. TR - 701, July 1994.

[4] W. Meier and O. Staffelbach, "Fast correlation attacks on certain stream ciphers," *J. Cryptol.*, vol. 1, no. 3, pp. 159–176, 1989.

[5] Y. Tarannikov, "New constructions of resilent Boolean function with maximum nonlinearity," *Lecture Notes in Computer Science*, vol. 2355, pp. 66–77, 2001.

[6] R. Bialota and G. Kawa, "Modified alternating k generators," *Des. Codes Cryptography*, vol. 35, no. 2, pp. 159–174, 2005.

[7] K. Zeng, C. Yang, D. Wei, and T. R. N. Rao, "Pseudo-random bit generators in stream-cipher cryptography," *Computer*, 1991.

[8] E. Biham and O. Dunkelman, "Cryptanalysis of the A5/1 GSM stream cipher," in *INDOCRYPT '00: Proceedings of the First International Conference on Progress in Cryptology*, (London, UK), pp. 43–51, Springer-Verlag, 2000.

[9] B. Lohlein, "Attacks based on conditional correlations against the nonlinear filter generator," citeseer.ist.psu.edu/554481.html.

[10] O. Y. Shaked, "Cryptanalysis of the Bluetooth E0 cipher," citeseer.ist.psu.edu/744254.html.

[11] D. Coppersmith, H. Krawczyk, and Y. Mansour, "The shrinking generator," in *CRYPTO '93: Proceedings of the 13th annual international cryptology conference on Advances in cryptology*, (New York, NY, USA), pp. 22–39, Springer-Verlag New York, Inc., 1994.

[12] B. Gammel, R. Göttfert, and O. Kniffler, "Achterbahn-128/80: Design and analysis," in *SASC'2007: Workshop Record of The State of the Art of Stream Ciphers*, pp. 152–165, 2007.

[13] M. Hell, T. Johansson, and W. Meier, "Grain - a stream cipher for constrained environments," citeseer.ist.psu.edu/732342.html.

[14] A. Maximov, "Cryptanalysis of the "Grain" family of stream ciphers," in *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, (New York, NY, USA), pp. 283–288, ACM Press, 2006.

[15] A. Bogdanov, "Cryptanalysis of the KeeLoq block cipher." Cryptology ePrint Archive, Report 2007/055, 2007. http://eprint.iacr.org/.

[16] C. D. Canniere and B. Preneel, "TRIVIUM specifications," citeseer.ist.psu.edu/734144.html.

[17] B. Gittins, H. A. Landman, S. O'Neil, and R. Kelson, "A presentation on VEST hardware performance, chip area measurements, power consumption estimates and benchmarking in relation to the aes, sha-256 and sha-512." Cryptology ePrint Archive, Report 2005/415, 2005. http://eprint.iacr.org/.

[18] B. M. Gammel, R. Göttfert, and O. Kniffler, "An NLFSR-based stream cipher," in *ISCAS*, 2006.

[19] B. Preneel, "A survey of recent developments in cryptographic algorithms for smart cards," *Comput. Networks*, vol. 51, no. 9, pp. 2223–2233, 2007.

[20] A. Canteaut, "Open problems related to algebraic attacks on stream ciphers," in *WCC*, pp. 120–134, 2005.

[21] J. Mykkeltveit, "Nonlinear recurrences and arithmetic codes," *Information and Control*, vol. 33, no. 3, pp. 193–209, 1977.

[22] J. Mykkeltveit, M.-K. Siu, and P. Tong, "On the cycle structure of some nonlinear shift register sequences," *Information and Control*, vol. 43, no. 2, pp. 202–215, 1979.

[23] C. A. Ronce, *Feedback Shift Registers*, vol. 169. 1984.

[24] M. J. B. Robshaw, *On Binary Cequences with Certain Properties*. Ph.D. Thesis, University of London, 1992.

[25] D. Linardatos and N. Kalouptsidis, "Synthesis of minimal cost nonlinear feedback shift registers," *Signal Process.*, vol. 82, no. 2, pp. 157–176, 2002.

[26] A. Ahmad, M. J. Al-Mushrafi, and S. Al-Busaidi, "Design and study of a strong crypto-system model for e-commerce," in *ICCC '02: Proceedings of the 15th international conference on Computer communication*, (Washington, DC, USA), pp. 619–630, International Council for Computer Communication, 2002.

[27] J. S. I. Janicka-Lipska, "Boolean feedback functions for full-length nonlinear shift registers," *Telecommunications and Informatioin Technology*, vol. 5, pp. 28–29, 2004.

[28] R. Bryant, "Graph-based algorithms for Boolean function manipulation," *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, pp. 677–691, August 1986.

[29] M. Luby, *Pseudorandomness and Cryptographic Applications*. Princeton, NJ, USA: Princeton University Press, 1994.

[30] J. Burch, E. Clarke, K. McMillan, D. Dill, and L. Hwang, "Symbolic Model Checking: 10^{20} States and Beyond," in *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, (Washington, D.C.), pp. 1–33, IEEE Computer Society Press, 1990.

[31] G. D. Hachtel and F. Somenzi, *Logic Synthesis and Verification Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

[32] F. S. Annexstein, "Generating de bruijn sequences: An efficient implementation," *IEEE Trans. Comput.*, vol. 46, no. 2, pp. 198–200, 1997.

[33] E. Dubrova and M. Teslenko, "Compositional properties of Random Boolean Networks," *Physical Review E*, vol. 71, p. 056116, May 2005.

[34] J. Burch, E. Clarke, D. E. Long, K. McMillan, and D. Dill, "Symbolic Model Checking for sequential circuit verification," *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, pp. 401–442, April 1994.

[35] S. M. Reddy, W. Kunz, and D. K. Pradhan, "Novel verification framework combining structural and OBDD methods in a synthesis environment," in *Proceedings of the 32th ACM/IEEE Design Automation Conference*, (San Francisco), pp. 414–419, June 1995.

[36] P. F. Williams, A. Biere, E. M. Clarke, and A. Gupta, "Combining decision diagrams and SAT procedures for efficient symbolic model checking," in *Computer Aided Verification (CAV'00)*, (Chicago, IL), pp. 125–138, Springer-Verlag, July 2000.