Improved Transformation Algorithms for Generalized Galois NLFSRs

Ge Yao*
School of Computing and Information Systems
The University of Melbourne
Melbourne, Australia
gyao1@student.unimelb.edu.au

Udaya Parampalli†
School of Computing and Information Systems
The University of Melbourne
Melbourne, Australia
udaya@unimelb.edu.au

Abstract

The transformation algorithm for Nonlinear Feedback Shift Registers (NLFSRs) convents NLFSRs between Fibonacci and Galois configurations. Up to now, three types of Galois NLFSRs namely Type-I, Type-II and Type-III Galois NLFSRs have been discovered to be equivalent to Fibonacci NLFSRs in existing algorithms. We discover a new type of Galois NLFSR (Type-IV) that can be transformed to Fibonacci NLFSR and propose a Fibonacci-to-Galois and a Galois-to-Fibonacci transformation algorithms for this new type. The proposed algorithms are based on a compensation method. Moreover, this method is adopted to propose a transformation algorithm for Type-I Galois NLFSRs and improve the transformation algorithms for the Type-II and Type-III Galois NLFSRs. No matter what the output function is the output sequences are the same before and after transformation.

1 Introduction

An n-bit NLFSR consists of n binary storage elements. We refer to each storage element as a stage represented by $x_i, i \in [0, n-1]$. The bit values stored in all the stages constitute

*The work of Ge Yao is funded by the China Scholarship Council - University of Melbourne PhD Scholarship.

†The work of U. Parampalli is supported in part by the Communications Sensing and Coding Research Network of the University of Melbourne's 2014 International Research and Research Training Fund (IRRTF).

the internal state of the NLFSR denoted by $X^t = \{x_0^t, x_1^t, \ldots, x_{n-1}^t\}$ at clock t. At the next clock, the internal state is updated by shifting the bit values one stage to the right and the last stage is updated by a feedback function $f_{n-1} = f(x_0, \ldots, x_{n-1})$ which is a Boolean function. Therefore, the internal state at $t+1$ contains $x_i^{t+1} = x_{i+1}^t, i \in [0, n-2]$ and $x_{n-1}^{t+1} = f(x_0^t, \ldots, x_{n-1}^t)$. This is the definition of NLFSR in Fibonacci configuration. There is another configuration for NLFSR, namely Galois NLFSR. In the Fibonacci configuration, the feedback is only applied to the last stage x_{n-1}. In the Galois configuration, the feedback can be applied to arbitrary stage $x_i, i \in [0, n-1]$. In Figure 1, an example of 4-bit Fibonacci NLFSR is given. The feedback f_3 is only applied to the last stage x_3. An example of 4-bit Galois NLFSR is also presented. The feedback f_2 is applied to x_2 and the feedback f_3 is applied to x_3.

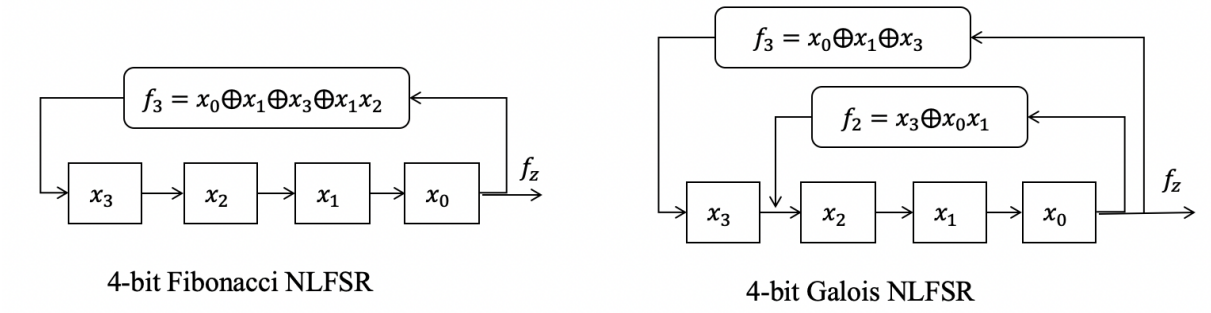

Figure 1: Examples of Fibonacci and Galois NLFSRs.

NLFSR is one of the building blocks of modern stream ciphers. Usually, Fibonacci NLFSR is used to design stream ciphers. For example, in the Grain family stream ciphers [1, 2, 3], a Fibonacci NLFSR is combined with a LFSR and a output function to generate keystream bits. Several following works [4, 5, 6] also adopt this structure. In other stream cipher such as Trivium [7] and Espresso [8], only Galois NLFSR is employed.

The theory of NLFSR has not been thoroughly developed, especially the analysis of Galois NLFSR is not well conducted. One of the research direction is the transformation between Fibonacci and Galois NLFSRs. In [9], the authors are the first to study the equivalence relation between these two kinds of NLFSRs and present the linear transformation of Galois NLFSRs defined by

$$\begin{aligned} &f_{n-1} = f(x_0, \ldots, x_{n-1}) \\ &f_i = x_{i+1} \oplus c_i \cdot f(x_0, \ldots, x_{n-1}) \quad for \quad i = 0, \ldots, n-2. \end{aligned} \tag{1}$$

where $c_i \in 0, 1$.

We refer to this kind of Galois NLFSRs as Type-I Galois NLFSR. In [10], the transformation from Fibonacci to Uniform Galois NLFSR is proposed. We refer Uniform Galois NLFSRs as Type-II Galois NLFSR.

$$\begin{aligned} &f_i = x_{(i+1) \mod n} \oplus g_i(x_0, \ldots, x_\tau) \quad for \quad i = \tau, \ldots, n-1 \\ &f_i = x_{i+1} \quad for \quad i = 0, \ldots, \tau - 1. \end{aligned} \tag{2}$$

where $\tau \in [0, n-1]$ and g_i represents the combination of remaining monomials in function f_i except the $x_{(i+1) \mod n}$ throughout this paper.

The following work [11] present the method to compute initial state of the Type-II Galois NLFSR based on the initial state of the original Fibonacci NLFSR. Later, a slightly generalized type of Galois NLFSR which we refer as Type-III Galois NFLSR is proved to be able to transformed to Fibonacci NLFSR in [12].

$$f_i = x_{(i+1) \mod n} \oplus g_i(x_0, \ldots, x_i) \quad for \quad i = 0, \ldots, n-1. \tag{3}$$

The method of finding matching initial state is also presented. In [13], the transformation between different Galois NLFSRs is studied. The authors prove that if the certain constrains are satisfied during the transformation then a Galois NLFSR can be transformed to a different one. Later, in [14], the authors show that several constrains in [13] can be relaxed. In [15], a new transformation algorithm based on semi-tensor product of matrices is proposed. An example given in the paper shows that this algorithm works on Type-II Galois NLFSRs.

Existing algorithms focus on either transforming Fibonacci NLFSR to more Galois NLFSR or finding more categories of Galois NFLSR to be able to transformed to Fibonacci one. But one thing has been ignored is the output function. Existing algorithms only studied equivalence of output sequences generated by stage x_0 in the NLFSR. Whether the sequence generated by other stages is not known. This issue causes problems when the transformation algorithm is applied on NLFSR used in stream ciphers. The output sequence of stream ciphers are usually generated by an output function which takes multiple stages of the NLFSR as input. After transformation, the output sequences of the stream cipher before and after transformation may not be equivalent.

In this paper, we discover that a new type of Galois NLFSR namely Type-IV Galois NLFSR in (4) is equivalent to a Fibonacci NLFSR. We develop a compensation method to build a relation of the internal states of the NLFSR before and after transformation. Then we propose a Fibonacci-to-Type-IV transformation algorithm to convert Fibonacci NLFSRs to Type-IV Galois NLFSRs and a Type-IV-to-Fibonacci transformation algorithm to convert Type-IV Galois NLFSRs to Fibonacci NLFSRs. No matter what the output function of the original NLFSR is, the proposed algorithms give a step to construct corresponding output function for the transformed NLFSR. We also apply the compensation method on other types of Galois NLFSR. We propose a Type-I-to-Fibonacci algorithm for the Type-I case and improve the transformation algorithms for the Type-II and Type-III Galois NLFSRs.

$$\begin{aligned} &f_{n-1} = x_0 \oplus g_{n-1}(x_0, \ldots, x_{n-1}) \\ &f_{n-2} = x_{n-1} \\ &f_i = x_{i+1} \oplus g_i(x_{i+2}, \ldots, x_{n-1}) \quad for \quad i = 0, \ldots, \tau - 1. \end{aligned} \tag{4}$$

where $\tau \in [0, n-1]$.

2 Notations

dep(·) is the dependence list of a monomial or a Boolean function. For example, the dependence list of a monomial $m = x_2x_3$ is $dep(m) = [2, 3]$ and the dependence list of a Boolean function $g = x_0 \oplus x_2x_3$ is $dep(g) = [0, 2, 3]$. If the indexes in the dependence list of m or g are decreased by i, we denote it as $\boldsymbol{m|_{-i}}$ or $\boldsymbol{g|_{-i}}$. Similarly, if the indexes are increased by $i \geq 0$, we denote it as $\boldsymbol{m|_{+i}}$ or $\boldsymbol{g|_{+i}}$. For example, $m|_{-1} = x_1x_2$ and $dep(m|_{-1}) = [1, 2]$, $m|_{+1} = x_3x_4$ and $dep(m|_{+1}) = [3, 4]$.

The equivalence of two NLFSRs is defined as

Definition 1. [12] Two NLFSRs are ***equivalent*** if their sets of output sequences are equal.

A formal description of transformation between Fibonacci and Galois NLFSR is first introduced in [10]. The transformation is done by shifting monomials in the feedback function f_{n-1} of the Fibonacci NLFSR to other feedback functions.

Definition 2. [10] Let f_a and f_b be feedback functions of stage x_a and x_b of an n-bit NLFSR, where $a, b \in [0, n-1]$. The ***shift*** operation moves a monomial m from f_a to f_b. As a result, the feedback function f_a becomes $f_a \oplus m$ and

$$\begin{aligned} f_b &= f_b \oplus m|_{-(a-b)} \quad if \quad a \geq b \\ f_b &= f_b \oplus m|_{+(a-b)} \quad if \quad a < b. \end{aligned}$$

In this paper, we define a backwards shift operation as

Definition 3. Let f_a and f_b be feedback functions of stage x_a and x_b of an n-bit NLFSR, where $a, b \in [0, n-1]$. The ***backwards shift*** operation moves a monomial m from f_a to f_b. As a result, the feedback function f_a becomes $f_a \oplus m$ and

$$\begin{aligned} f_b &= f_b \oplus m|_{+(n-1-a+b)} \quad if \quad a \geq b \\ f_b &= f_b \oplus m|_{-(n-1-a+b)} \quad if \quad a < b. \end{aligned}$$

As discussed in Section 1, it is important to construct the output function for the transformed NLFSR. Therefore, we add the process of constructing output in the transformation algorithm. Formally, the ***transformation algorithm*** consists of:

1. Construct the feedback functions for the transformed NLFSR;
2. Construct the output function for the transformed NLFSR;
3. Compute the set of initial states of the transformed NLFSR according to the set of initial sates of the original NLFSR.

Different from the transformation algorithm in [10] and [12], all the proposed transformation algorithms consist the above processes.

In order to construct the output function for the transformed NLFSR, we need to build a relation of the internal states of the NLFSR before and after transformation. To this end, we introduce the compensation list and the compensate operation.

Definition 4. Given an n-bit NLFSR, suppose a monomial m is backwards shifted from f_{n-1} to $f_\tau, \tau \in [0, n-3]$, the ***compensation list*** is constructed as $C = [c_0, \ldots, c_{n-1}] = [m, m|_{+1}, \ldots, m|_{+\tau}, 0, \ldots, 0]$. Suppose a monomial m is backwards shifted from f_τ to f_{n-1}, then the ***compensation list*** is constructed as $C = [c_0, \ldots, c_{n-1}] = [m|_{-\tau-1}, m|_{-\tau}, \ldots, m|_{-1}, 0, \ldots, 0]$. The combination of two compensation lists C_{m_0} and C_{m_1} is defined as bitwise xor the elements in two lists denoted as $C = C_{m_0} \oplus C_{m_1}$.

For example, given a 4-bit NLFSR, if the monomial $m = x_2$ is backwards shifted from f_3 to f_1, then the compensation list is $C_m = [x_2, x_3, 0, 0]$. If the monomial $m = x_3$ is backwards shifted from f_0 to f_3, then the compensation list is $C_m = [x_2, 0, 0, 0]$.

Definition 5. Given a compensation list C and a Boolean function f or a monomial m, the ***compensate*** operation replaces each $x_i, i \in [0, n-1]$ by $x_i \oplus c_i$ in ascending order or descending order of the index i with or without iteration, where c_i is the i-th element of C. We denote the compensated result as $\overline{f}$ or $\overline{m}$.

For example, we use a compensation list $C = [x_2, x_3, 0, 0]$ to compensate $m = x_0x_1$ iteratively in ascending order. The compensating operation first replace x_0 by $x_0 \oplus x_2$ in m and m becomes $(x_0 \oplus x_2)x_1$ then replace the x_1 by $x_1 \oplus x_3$ and m becomes $\overline{m} = (x_0 \oplus x_2)(x_1 \oplus x_3)$.

$$m = x_0x_1 \xrightarrow{replace\, x_0} (x_0 \oplus x_2)x_1 \xrightarrow{replace\, x_1} \overline{m} = (x_0 \oplus x_2)(x_1 \oplus x_3)$$

3 Proposed Transformation Algorithms

In this section, we first analyse the differences in internal states of the original NLFSRs and the transformed Type-IV Galois NLFSR. We find that there is a connection between the differences and the monomials shifted during the transformation. Based on the connection, we establish a relation between the internal states of the NLFSR before and after transformation in Theorem 6.

Theorem 6. *Given an n-bit Fibonacci NLFSR with an output function f_z and an initial value X^0, r monomials $m_0, m_1, \ldots, m_{r-1}$ are backwards shifted from f_{n-1} to feedback functions $f_{\tau_0}, \ldots, f_{\tau_{r-1}}, 0 \le \tau_0 < \tau_1 < \ldots < \tau_{r-1} \le n-2$ respectively. If the resulted Galois NLFSR is a Type-IV NLFSR, the initial state is $\hat{X}^0 = \{\hat{x}_i^0 = x_i^0 \oplus c_i(X^0), i \in [0, n-1]\}$ and feedback functions are compensated by the compensation list C iteratively from $i = 0$ to $i = n-1$ and, then the internal states of the NLFSR before and after transformation satisfy*

$$\begin{aligned} \hat{x}_i^t &= x_i^t \quad for \quad i \in [\tau_{r-1}+1, n-1] \\ \hat{x}_i^t &= x_i^t \oplus c_i(X^t) \quad for \quad i \in [0, \tau_{r-1}]. \end{aligned} \tag{5}$$

or

$$\begin{aligned} x_i^t &= \hat{x}_i^t \quad for \quad i \in [\tau_{r-1}+1, n-1] \\ x_i^t &= \hat{x}_i^t \oplus \overline{c_i}(\hat{X}^t) \quad for \quad i \in [0, \tau_{r-1}]. \end{aligned} \tag{6}$$

Proof. According to Definition 4, the compensation list for each shifted monomial is

$$\begin{aligned}
C_{m_0} &= [m_0, \ldots, m_0|_{+\tau_0}, 0, \ldots, 0] \\
C_{m_1} &= [m_1, \ldots, m_1|_{+\tau_1}, 0, \ldots, 0] \\
&\ldots \\
C_{m_{r-1}} &= [m_r, \ldots, m_r|_{+\tau_r}, 0, \ldots, 0].
\end{aligned}$$

Then the combined compensation list is $C = C_{m_0} \oplus C_{m_1} \oplus \ldots \oplus C_{m_{r-1}}$.

Without loss of generality, we suppose only two monomials m_0 and m_1 are shifted to f_{τ_0} and $f_{\tau_1}, \tau_0 < \tau_1$. The compensation lists are

$$\begin{aligned}
C_{m_0} &= [m_0, \ldots, m_0|_{+\tau_0}, 0, \ldots, 0] \\
C_{m_1} &= [m_1, \ldots, m_1|_{+\tau_1}, 0, \ldots, 0]
\end{aligned}$$

Then the combination of above lists is $C = C_{m_0} \oplus C_{m_1} = [c_0, \ldots, c_{n-1}]$. Specifically,

$$\begin{aligned}
c_i &= m_0|_{+i} \oplus m_1|_{+i} \qquad i \in [0, \tau_0] \\
c_i &= m_1|_{+i} \qquad i \in [\tau_0 + 1, \tau_1] \\
c_i &= 0 \qquad i \in [\tau_1 + 1, n-1]
\end{aligned}$$

In this theorem, the feedback functions of the transformed NLFSR are compensated by C. The feedback functions of the Fibonacci NLFSR and the transformed Galois NLFSR are

$$\begin{array}{lcl}
f_{n-1} = x_0 \oplus g_{n-1} & & f_{n-1} = x_0 \oplus \overline{g_{n-1} \oplus m_1 \oplus m_0} \\
\quad \ldots & & \quad \ldots \\
f_{\tau_1+1} = x_{\tau_1+2} & & f_{\tau_1+1} = x_{\tau_1+2} \\
f_{\tau_1} = x_{\tau_1+1} & & f_{\tau_1} = x_{\tau_1+1} \oplus \overline{m_1|_{+\tau_1+1}} \\
f_{\tau_1-1} = x_{\tau_1} & f_{n-1} \xrightarrow{m_1} f_{\tau_1} & f_{\tau_1-1} = x_{\tau_1} \\
\quad \ldots & f_{n-1} \xrightarrow{m_0} f_{\tau_0} & \quad \ldots \\
\quad \ldots & \Longrightarrow & \quad \ldots \\
f_{\tau_0+1} = x_{\tau_0+2} & & f_{\tau_0+1} = x_{\tau_0+2} \\
f_{\tau_0} = x_{\tau_0+1} & & f_{\tau_0} = x_{\tau_0+1} \oplus \overline{m_0|_{+\tau_0+1}} \\
f_{\tau_0-1} = x_{\tau_0} & & f_{\tau_0-1} = x_{\tau_0} \\
\quad \ldots & & \quad \ldots \\
f_0 = x_1, & & f_0 = x_1,
\end{array}$$

where $\overline{g_{n-1} \oplus m_1 \oplus m_0}$ is $g_{n-1} \oplus m_1 \oplus m_0$ compensated by C.

Now we prove the relation in (5) holds by induction. Suppose the relationship holds

at clock $t = k$, we have

$$\begin{aligned}
&\hat{x}^k_{n-1} = x^k_{n-1}\\
&\quad\cdots\\
&\hat{x}^k_{\tau_1+1} = x^k_{\tau_1+1}\\
&\hat{x}^k_{\tau_1} = x^k_{\tau_1} \oplus m_1|_{+\tau_1}(X^k)\\
&\quad\cdots\\
&\hat{x}^k_{\tau_0+1} = x^k_{\tau_0+1} \oplus m_1|_{+\tau_0+1}(X^k)\\
&\hat{x}^k_{\tau_0} = x^k_{\tau_0} \oplus m_0|_{+\tau_0}(X^k) \oplus m_1|_{+\tau_0}(X^k)\\
&\quad\cdots\\
&\hat{x}^k_0 = x^k_0 \oplus m_0(X^k) \oplus m_1(X^k)
\end{aligned} \tag{7}$$

At next clock $t = k + 1$ the internal states of the Fibonacci NLFSR and the Galois NLFSR are computed according to their feedback functions

$$\begin{aligned}
&x^{k+1}_{n-1} = x^k_0 \oplus g_{n-1}(X^k) && \hat{x}^{k+1}_{n-1} = \hat{x}^k_0 \oplus \overline{g_{n-1} \oplus m_1 \oplus m_0}(\hat{X}^k)\\
&x^{k+1}_{n-2} = x^k_{n-1} && \hat{x}^{k+1}_{n-2} = \hat{x}^k_{n-1}\\
&\quad\cdots && \quad\cdots\\
&x^{k+1}_{\tau_1+1} = x^k_{\tau_1+2} && \hat{x}^{k+1}_{\tau_1+1} = \hat{x}^k_{\tau_1+2}\\
&x^{k+1}_{\tau_1} = x^k_{\tau_1+1} && \hat{x}^{k+1}_{\tau_1} = \hat{x}^k_{\tau_1+1} \oplus \overline{m_1|_{+\tau_1+1}}(\hat{X}^k)\\
&x^{k+1}_{\tau_1-1} = x^k_{\tau_1} && \hat{x}^{k+1}_{\tau_1-1} = \hat{x}^k_{\tau_1}\\
&\quad\cdots && \quad\cdots\\
&x^{k+1}_{\tau_0+1} = x^k_{\tau_0+2} && \hat{x}^{k+1}_{\tau_0+1} = \hat{x}^k_{\tau_0+2}\\
&x^{k+1}_{\tau_0} = x^k_{\tau_0+1} && \hat{x}^{k+1}_{\tau_0} = \hat{x}^k_{\tau_0+1} \oplus \overline{m_0|_{+\tau_0+1}}(\hat{X}^k)\\
&x^{k+1}_{\tau_0-1} = x^k_{\tau_0} && \hat{x}^{k+1}_{\tau_0-1} = \hat{x}^k_{\tau_0}\\
&\quad\cdots && \quad\cdots\\
&x^{k+1}_0 = x^k_1. && \hat{x}^{k+1}_0 = \hat{x}^k_1.
\end{aligned} \tag{8}$$

We replace $\hat{x}^k_i, i \in [0, n-1]$ in (8) by (7)

$$
\begin{aligned}
&\hat{x}_{n-1}^{k+1} = x_0^k \oplus m_0(X^k) \oplus m_1(X^k) \oplus \overline{g_{n-1} \oplus m_1 \oplus m_0}(\hat{X}^k)\\
&\hat{x}_{n-2}^{k+1} = x_{n-1}^k\\
&\quad \cdots\\
&\hat{x}_{\tau_1+1}^{k+1} = x_{\tau_1+2}^k\\
&\hat{x}_{\tau_1}^{k+1} = x_{\tau_1+1}^k \oplus \overline{m_1|_{+\tau_1+1}}(\hat{X}^k)\\
&\hat{x}_{\tau_1-1}^{k+1} = x_{\tau_1}^k \oplus m_1|_{+\tau_1}(X^k)\\
&\quad \cdots\\
&\hat{x}_{\tau_0+1}^{k+1} = x_{\tau_0+2}^k \oplus m_1|_{+\tau_0+2}(X^k)\\
&\hat{x}_{\tau_0}^{k+1} = x_{\tau_0+1}^k \oplus m_1|_{+\tau_0+1}(X^k) \oplus \overline{m_0|_{+\tau_0+1}}(\hat{X}^k)\\
&\hat{x}_{\tau_0-1}^{k+1} = x_{\tau_0}^k \oplus m_0| + \tau_0(X^k) \oplus m_1|_{+\tau_0}(X^k)\\
&\quad \cdots\\
&\hat{x}_0^{k+1} = x_1^k \oplus m_0|_{+1}(X^k) \oplus m_1|_{+1}(X^k).
\end{aligned}
\tag{9}
$$

From (8) and (9), we have $\hat{x}_i^{k+1} = x_i^{k+1}$ for $i \in [\tau_1+1, n-2]$, and $\hat{x}_i^{k+1} = x_i^{k+1} \oplus c_i(X^{k+1})$. For $\hat{x}_{n-1}^{k+1}$, we need to prove that $\overline{g_{n-1} \oplus m_1 \oplus m_0}(\hat{X}^k) = g_{n-1} \oplus m_0 \oplus m_1(X^k)$. To this end, we suppose that $\tau_0 = 1$ and $\tau_1 = 2$, then $C = [m_0 \oplus m_1, m_0|_{+1} \oplus m_1|_{+1}, m_1|_{+2}, 0, \ldots, 0]$. We get

$$
\begin{aligned}
\overline{g_{n-1} \oplus m_1 \oplus m_0}(\hat{X}^k) &= g_{n-1} \oplus m_0 \oplus m_1(\hat{x}_1^k \oplus \overline{c_1}(\hat{X}^k), \hat{x}_2^k \oplus c_2(\hat{X}^k), \hat{x}_3^k, \ldots, \hat{x}_{n-1}^k)\\
&= g_{n-1} \oplus m_0 \oplus m_1(\hat{x}_1^k \oplus c_1(\hat{x}_2^k \oplus c_2(\hat{X}^k), \hat{x}_3^k, \ldots, \hat{x}_{n-1}^k),\\
&\quad \hat{x}_2^k \oplus c_2(\hat{X}^k), \hat{x}_3^k, \ldots, \hat{x}_{n-1}^k)\\
&= g_{n-1} \oplus m_0 \oplus m_1(\hat{x}_1^k \oplus c_1(X^k), x_2^k, x_3^k, \ldots, x_{n-1}^k)\\
&= g_{n-1} \oplus m_0 \oplus m_1(x_1^k, x_2^k, x_3^k, \ldots, x_{n-1}^k)\\
&= g_{n-1} \oplus m_0 \oplus m_1(X^k).
\end{aligned}
\tag{10}
$$

For general cases when $\tau_0 > 1$ and $\tau_1 > 2$, due to the iterative replacement in the compensation operation, we can always get $\overline{g_{n-1} \oplus m_1 \oplus m_0}(\hat{X}^k) = g_{n-1} \oplus m_0 \oplus m_1(X^k)$. Therefore, we have $\hat{x}_{n-1}^{k+1} = x_0^k \oplus g_{n-1}(X^k) = x_{n-1}^{k+1}$.

For $\hat{x}_{\tau_1}^{k+1}$, since $dep(m_1|_{+\tau_1+1}) \subseteq [\tau_1+1, n-1]$, there is no replacement of any tap, then $\overline{m_1|_{+\tau_1+1}}(\hat{X}^k) = m_1|_{+\tau_1+1}(\hat{X}^k) = m_1|_{+\tau_1+1}(\hat{x}_{\tau_1+1}^k, \ldots, \hat{x}_{n-1}^k) = m_1|_{+\tau_1+1}(X^k)$, therefore, $\hat{x}_{\tau_1}^{k+1} = x_{\tau_1}^{k+1} \oplus m_1|_{+\tau_1+1}(X^k) = x_{\tau_1}^{k+1} \oplus m_1|_{+\tau_1}(X^{k+1}) = x_{\tau_1}^{k+1} \oplus c_{\tau_1}(X^{k+1})$.

For $\hat{x}_{\tau_0}^{k+1}$, since $dep(m_0|_{+\tau_0+1}) \subseteq [\tau_0+2, n-1]$, similar to the proof of $\overline{g_{n-1} \oplus m_1 \oplus m_0}(\hat{X}^k) = g_{n-1} \oplus m_0 \oplus m_1(X^k)$, the iterative replacement in the compensation of $m_0|_{+\tau_0+1}$ assures that $\overline{m_0|_{+\tau_0+1}}(\hat{X}^k) = m_0|_{+\tau_0+1}(X^k)$. Then we have $\hat{x}_{\tau_0}^{k+1} = x_{\tau_0}^{k+1} \oplus m_1|_{+\tau_0+1}(X^k) \oplus m_0|_{+\tau_0+1}(X^k) = x_{\tau_0}^{k+1} \oplus m_1|_{+\tau_0}(X^{k+1}) \oplus m_0|_{+\tau_0}(X^{k+1}) = x_{\tau_0}^{k+1} \oplus c_{\tau_0}(X^{k+1})$.

Therefore, the relation in (5) holds for clock $t = k+1$. If the initial state of the transformed NLFSR is computed the same way, then the relationship is valid for every clock.

Similarly, the relation in (6) can be proved by induction. □

Based on Theorem 6, we propose the Fibonacci-to-Type-IV and the Type-IV-to-Fibonacci transformation algorithms.

3.1 Fibonacci-to-Type-IV

Given an n-bit Fibonacci NLFSR with an output function f_z and an initial value X^0, the transformation process is:

Step 1: Shift r monomials $m_0, m_1, \ldots, m_{r-1}$ backwards from f_{n-1} to feedback functions $f_{\tau_0}, \ldots, f_{\tau_{r-1}}, 0 \leq \tau_0 < \tau_1 < \ldots < \tau_{r-1} \leq n-2$ respectively, the resulted feedback functions satisfy the condition of Type-IV Galois NLFSRs in (4);

Step 2: The output function and feedback functions are compensated by the compensation list C iteratively from $i = 0$ to $i = n-1$;

Step 3: The initial state is computed as $\hat{x}_i^0 = x_i^0 \oplus c_i(X^0), i \in [0, n-1]$.

Correctness: Since the output function f_z is compensated in Step 2, any tap $x_i, i \in [0, \tau_{r-1}]$ in f_z is compensated by C. Since the compensation is done iteratively, the x_i in c_i is also compensated. Therefore, at each clock t, the bit value of that tap in $\overline{f_z}$ is equal to $\hat{x}_i^t \oplus \overline{c_i}(\hat{X}^t)$ which is equal to x_i^t according to (6). Hence, the output function $\overline{f_z}$ generate same bit as the original function f_z does at each clock.

To be noted that the r monomials can be chosen from the monomials in f_{n-1} or constructed by the combination of stages of the NLFSR. Due to the limitation of feedback functions in a Type-IV Galois NLFSR (4), the number of possible positions to which monomials of a Fibonacci NLFSR can be shifted is limited. Given r monomials $m_0, m_1, \ldots, m_{r-1}$ to be backwards shifted in an n-bit Fibonacci NLFSR, the lowest position where each monomial m_j can be shifted to is calculated as

$$p_j = n - 2 - max(dep(m_j)). \tag{11}$$

The pseudocode is

Algorithm 1 Type-IV_Shifting_Position

```
1: procedure ShiftingPosition(n, m_0, ..., m_{r-1})
2:     for j ← 0, r − 1 do
3:         τ_j ← n − 2 − max(dep(m_j))
```

The pseudocode for the Fibonacci-to-Type-IV transformation algorithm is

Algorithm 2 Fibonacci-to-Type-IV

procedure FTTIV(n, $f_0, \ldots, f_{n-1}$, f_z, $m_0, \ldots, m_{r-1}$, $\tau_0, \ldots, \tau_{r-1}$, X^0)
  $C \leftarrow [0, \ldots, 0]$
  for $j \leftarrow 0, r-1$ **do**
    $f_{n-1} \leftarrow f_{n-1} \oplus m_j$ ▷ shift monomial m_j from f_{n-1}
    $m_j|_{+(\tau_j+1)} \leftarrow m_j$ with $dep(m_j) + (\tau_j + 1)$
    $f_{\tau_j} \leftarrow f_{\tau_j} \oplus m_j|_{+(\tau_j+1)}$ ▷ shift monomial m_j to f_{τ_j}
    $C_{m_j} \leftarrow [0, \ldots, 0]$
    for $k \leftarrow 0, \tau_j$ **do**
      $m_j|_{+k} \leftarrow m_j$ with $dep(m_j) + k$
      $C_{m_j}[k] \leftarrow m_j|_{+k}$
    $C \leftarrow C \oplus C_{m_j}$ ▷ combine all the compensation lists
  for $i \leftarrow 0, n-1$ **do**
    if $c_i \neq 0$ **then** ▷ c_i is the i-th element in C
      $f_z \leftarrow f_z$ with x_i replaced by $x_i \oplus c_i$ ▷ compensate f_z by C iteratively
      for $j \leftarrow 0, n-1$ **do**
        $f_j \leftarrow f_j$ with x_i replaced by $x_i \oplus c_i$ ▷ compensate f_j by C iteratively
  for $i \leftarrow 0, n-1$ **do**
    $\hat{x}_i^0 = x_i^0 \oplus c_i(X^0)$
    $\hat{X}^0[i] \leftarrow \hat{x}_i^0$ ▷ calculate initial state for the transformed NLFSR
  return $f_0, \ldots, f_{n-1}$, f_z, $\hat{X}^0$

For example, a 7-bit Fibonacci NLFSR is transformed to a Type-IV Galois NLFSR:

$$\begin{aligned} f_6 &= x_0 \oplus x_4x_5 \oplus x_1 \oplus x_2 \oplus x_1 \\ f_5 &= x_6 \\ f_4 &= x_5 \\ f_3 &= x_4 \\ f_2 &= x_3 \\ f_1 &= x_2 \\ f_0 &= x_1, \end{aligned} \qquad \begin{aligned} &f_6 \xrightarrow{m_1=x_1} f_{\tau_1=3} \\ &f_6 \xrightarrow{m_0=x_2\oplus x_1} f_{\tau_0=1} \end{aligned} \qquad \begin{aligned} f_6 &= x_0 \oplus x_4x_5 \\ f_5 &= x_6 \\ f_4 &= x_5 \\ f_3 &= x_4 \oplus x_5 \\ f_2 &= x_3 \\ f_1 &= x_2 \oplus x_3 \\ f_0 &= x_1. \end{aligned}$$

The compensation list $C = [x_2, x_3, x_3, x_4, 0, 0, 0]$. Suppose the output function of the Fibonacci NLFSR is $f_z = x_2 \oplus x_3$, then the output function for the Type-IV Galois NLFSR $\overline{f_z}$ is constructed as

$$f_z = x_2 \oplus x_3 \xrightarrow{replace\, x_2} (x_2 \oplus x_3) \oplus x_3 \xrightarrow{replace\, x_3} \overline{f_z} = (x_2 \oplus (x_3 \oplus x_4)) \oplus (x_3 \oplus x_4) = x_2$$

The initial states of the Type-IV Galois NLFSR is $\hat{X}^0 = \{\hat{x}_0^0, \hat{x}_1^0, \hat{x}_2^0, \hat{x}_3^0, \hat{x}_4^0, \hat{x}_5^0, \hat{x}_6^0\}$, where

$$\begin{aligned}
\hat{x}_6^0 &= x_6^0 \\
\hat{x}_5^0 &= x_5^0 \\
\hat{x}_4^0 &= x_4^0 \\
\hat{x}_3^0 &= x_3^0 \oplus x_4^0 \\
\hat{x}_2^0 &= x_2^0 \oplus x_3^0 \\
\hat{x}_1^0 &= x_1^0 \oplus x_3^0 \\
\hat{x}_0^0 &= x_0^0 \oplus x_2^0.
\end{aligned}$$

3.2 Type-IV-to-Fibonacci

Given an n-bit Type-IV Galois NLFSR defined in (4) with an output function f_z and an initial value X^0, all the monomials in g_i are shifted from $f_i, i \in [0, n-3]$ to f_{n-1}, the Galois NLFSR is transformed to a Fibonacci NLFSR in following steps:

Step 1: Let the combined compensation list $C = [0, \ldots, 0]$;

Step 2: For each g_i, starts from $i = n-3$, remove it from f_i and construct a compensation list $C_i = [g_i|_{-i-1}, g_i|_{-i}, \ldots, g_i|_{-1}, 0, \ldots, 0]$ and compute $C = C \oplus C_i$;

Step 3: Compensation: only use $C[i]$ to compensate the tap x_i in all the feedback functions, which means x_i is replaced by $x_i \oplus C[i]$;

Step 4: If $i \neq 0$, then set $i = i - 1$ and go back to Step 2, otherwise, go to Step 5;

Step 5: The final combined compensation list is C. Xor all the shifted monomials $g_i|_{-i-1}, i \in [0, n-3]$ to the feedback function of bit x_{n-1} to get the final feedback functions for the transformed NLFSR;

Step 6: The output function $\overline{f_z}$ is constructed by replacing $x_i, i \in [0, n-1]$ in f_z by $x_i \oplus c_i$, no iteration is needed in this compensation process;

Step 7: The initial value $\hat{X}^0 = \{\hat{x}_0^0, \ldots, \hat{x}_{n-1}\}$ is computed by compensating $X^0 = \{x_0^0, \ldots, x_{n-1}^0\}$ by C^0 iteratively starting from $i = 0$ to $i = n-1$.

The transformed NLFSR is a Fibonacci NLFSR and it generates the same sequence as the Galois NLFSR outputs.

Correctness: The transformation process is completely the reverse of the process in the Fibonacci-to-Type-IV transformation algorithm.

The pseudocode for this algorithm is

Algorithm 3 Type-IV-to-Fibonacci

procedure TIVTF(n, $f_0, \ldots, f_{n-1}$, f_z, X^0)
  $C_{g_i} \leftarrow [0, \ldots, 0]$
  for $i \leftarrow n-3, 0$ **do**
    $f_i \leftarrow f_i \oplus g_i$ ▷ shift monomials in g_i from f_i
    $C_{g_i} \leftarrow [0, \ldots, 0]$
    for $j \leftarrow 0, i+1$ **do**
      $g_i|_{-i-1+j} \leftarrow g_i$ with $dep(g_i) - i - 1 + j$
      $C_{g_i}[j] \leftarrow g_i|_{-i-1+j}$ ▷ construct compensation list
    $C \leftarrow C \oplus C_{g_i}$ ▷ combine all the compensation lists
    for $j \leftarrow 0, n-1$ **do**
      $f_j \leftarrow f_j$ with x_i replaced by $x_i \oplus c_i$ ▷ compensate only x_i
    $f_z \leftarrow f_z$ with x_{i+1} replaced by $x_{i+1} \oplus c_{i+1}$
  for $i \leftarrow 0, n-3$ **do**
    $f_{n-1} \leftarrow f_{n-1} \oplus g_i|_{-i-1}$ ▷ shift monomials to f_{n-1}
  for $i \leftarrow 0, n-1$ **do**
    $\overline{c_i} \leftarrow c_i$
    for $j \leftarrow n-1, 0$ **do**
      if $C[j] \neq 0$ **then**
        $\overline{c_i} \leftarrow \overline{c_i}$ with x_j replaced by $x_j \oplus c_j$ ▷ get compensated $\overline{c_i}$
    $\hat{x}_i^0 = x_i^0 \oplus \overline{c_i}(X^0)$
    $\hat{X}^0[i] \leftarrow \hat{x}_i^0$
return $f_0, \ldots, f_{n-1}$, f_z, $\hat{X}^0$

For example, the 7-bit Type-IV Galois NLFSR in the example in Section 3.1 is transformed back to the Fibonacci NLFSR:

$$\begin{array}{ll} f_6 = x_0 \oplus x_4x_5 & f_6 = x_0 \oplus x_4x_5 \oplus x_1 \oplus x_2 \oplus x_1 \\ f_5 = x_6 & f_5 = x_6 \\ f_4 = x_5 & f_4 = x_5 \\ f_3 = x_4 \oplus x_5 & f_3 = x_4 \\ f_2 = x_3 & f_2 = x_3 \\ f_1 = x_2 \oplus x_3 & f_1 = x_2 \\ f_0 = x_1, & f_0 = x_1. \end{array}$$

$f_3 \xrightarrow{g_3=x_5} f_6$, $f_1 \xrightarrow{g_1=x_3} f_6$

The compensation list $C = [x_2, x_3, x_3, x_4, 0, 0, 0]$. Suppose the output function of the Fibonacci NLFSR is $f_z = x_2 \oplus x_3$, then the output function for the Type-IV Galois NLFSR $\overline{f_z}$ is constructed as

$$f_z = x_2 \oplus x_3 \xrightarrow{replace\, x_2 and x_3} \overline{f_z} = (x_2 \oplus x_3) \oplus (x_3 \oplus x_4) = x_2 \oplus x_4$$

The initial states of the Type-IV Galois NLFSR is $\hat{X}^0 = \{\hat{x}_0^0, \hat{x}_1^0, \hat{x}_2^0, \hat{x}_3^0, \hat{x}_4^0, \hat{x}_5^0, \hat{x}_6^0\} = \{x_0^0 \oplus x_2^0, x_1^0 \oplus x_3^0 \oplus x_4^0, x_2^0 \oplus x_3^0 \oplus x_4^0, x_3^0 \oplus x_4^0, x_4^0, x_5^0, x_6^0\}$.

4 Improve Existing Algorithms

The compensation idea can be also adopted to construct output functions for the Type-I, Type-II and Type-III cases. We propose a Type-I-to-Fibonacci for the Type-I Galois NLFSR in Apendix 1, a Fibonacci-to-Type-II and Type-II-to-Fibonacci algorithms for the Type-II Galois NLFSR in Appendix 2. In Appendix 3, we present the Fibonacci-to-Type-III and Type-III-to-Fibonacci algorithms for the Type-III Galois NLFSR.

We compare our algorithm with existing algorithms at several aspects including method, construction of output function, coverage, reversibility and complexity.

Table 1: Comparison of Transformation Algorithms

Algorithms	Output	I	II	III	IV	FTG	GTF	Complexity
Dubrova et al. [10]	No	No	Yes	No	No	Yes	No	$\mathcal{O}(n)$
Lin et al. [12]	No	No	No	Yes	No	No	Yes	$\mathcal{O}(n^2)$
Lu et al. [15]	No	No	Yes	Yes	No	No	Yes	$\mathcal{O}(2^n)$
Our Algorithms	Yes	Yes	Yes	Yes	Yes	Yes	Yes	$\mathcal{O}(n)$ or $\mathcal{O}(n^2)$

In the table, the second column "Output" means the algorithm contains a step of constructing output function for the transformed NLFSR. The next three columns check if the algorithm covers the Type-I, Type-II, Type-III, Type-IV cases. The next two columns check if the algorithm transforms from Fibonacci to Galois and the vice versa. In the last column, n is the size of the NLFSR. The complexity of our proposed algorithms is $\mathcal{O}(n^2)$ except that the Fibonacci-to-Type-II algorithm has complexity of $\mathcal{O}(n)$ which is the same as the Dubrova et al. algorithm [10]. Compare to Lu et al. algorithm [15] which transforms Type-II and Type-III Galois NLFSRs to Fibonacci NLFSRs, our proposed algorithm has much better complexity. Overall, our proposed algorithm covers Type-II and Type-III cases and has lower complexity. More importantly, to authors' best knowledge, this is the first work to propose a transformation algorithm for Type-I Galois NLFSRs, an important open problem in the area. We discover a new type of Galois NLFSRs (Type-IV) that can be transformed to Fibonacci NLFSRs, which is a big step closer to identify all the transformable Galois NLFSRs. Another novel feature is that our proposed algorithms are the only ones that that constructs the output function for the transformed NLFSR. Besides, the proposed algorithms for Type-II, Type-III and Type-IV are reversible, both Fibonacci-to-Galois and Galois-to-Fibonacci algorithms are presented.

5 Conclusion

In this paper, we propose transformation algorithms for a new type (Type-IV) of Galois NLFSR. We develop an idea of compensating the functions of the NLFSR to help construct the output function of the transformed NLFSR in the proposed algorithms. We also apply this method to propose a transformation algorithm for Type-I case and improve the transformation algorithms for Type-II and Type-III cases. The next step is to identify

all the Galois NLFSRs that can be transformed to Fibonacci NLFSRs. The proposed transformation algorithms can be used to improve the throughput of NLFSR-based stream ciphers or adopted in attacks against such cipher.

Acknowledgements

We thank the reviewers for their insightful comments that helped us to improve the presentation.

References

[1] Hell, Martin, Thomas Johansson, and Willi Meier. "Grain: a stream cipher for constrained environments." International journal of wireless and mobile computing 2, no. 1 (2007): 86-93.

[2] Hell, Martin, Thomas Johansson, Alexander Maximov, and Willi Meier. "A stream cipher proposal: Grain-128." In 2006 IEEE International Symposium on Information Theory, pp. 1614-1618. IEEE, 2006.

[3] Ågren, Martin, Martin Hell, Thomas Johansson, and Willi Meier. "Grain-128a: a new version of Grain-128 with optional authentication." International Journal of Wireless and Mobile Computing 5, no. 1 (2011): 48-59.

[4] Armknecht, Frederik, and Vasily Mikhalev. "On lightweight stream ciphers with shorter internal states." In International Workshop on Fast Software Encryption, pp. 451-470. Springer, Berlin, Heidelberg, 2015.

[5] Mikhalev, Vasily, Frederik Armknecht, and Christian Müller. "On ciphers that continuously access the non-volatile key." IACR Transactions on Symmetric Cryptology (2016): 52-79.

[6] Ghafari, Vahid Amin, Honggang Hu, and Ying Chen. "Fruit-v2: ultra-lightweight stream cipher with shorter internal state." Int. Assoc. Cryptol. Res (IACR) (2016).

[7] De Cannière, Christophe. "Trivium: A stream cipher construction inspired by block cipher design principles." In International Conference on Information Security, pp. 171-186. Springer, Berlin, Heidelberg, 2006.

[8] Dubrova, Elena, and Martin Hell. "Espresso: A stream cipher for 5G wireless communication systems." Cryptography and Communications 9, no. 2 (2017): 273-289.

[9] Massey, J., and Ruey-Wen Liu. "Equivalence of nonlinear shift-registers." IEEE Transactions on Information Theory 10, no. 4 (1964): 378-379.

[10] Dubrova, Elena. "A transformation from the Fibonacci to the Galois NLFSRs." IEEE Transactions on Information Theory 55, no. 11 (2009): 5263-5271.

[11] Dubrova, Elena. "Finding matching initial states for equivalent NLFSRs in the Fibonacci and the Galois configurations." IEEE transactions on information theory 56, no. 6 (2010): 2961-2966.

[12] Zhiqiang, Lin. "The transformation from the Galois NLFSR to the Fibonacci configuration." In 2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies, pp. 335-339. IEEE, 2013.

[13] Dubrova, Elena. "An equivalence-preserving transformation of shift registers." In International Conference on Sequences and Their Applications, pp. 187-199. Springer, Cham, 2014.

[14] Li, Gefei, Yuval Yarom, and Damith Chinthana Ranasinghe. "Exploiting Transformations of the Galois Configuration to Improve Guess-and-Determine Attacks on NFSRs." IACR Cryptol. ePrint Arch. 2015 (2015): 1045.

[15] Lu, Jianquan, Meilin Li, Tingwen Huang, Yang Liu, and Jinde Cao. "The transformation between the Galois NLFSRs and the Fibonacci NLFSRs via semi-tensor product of matrices." Automatica 96 (2018): 393-397.

Appendix 1

The transformation algorithms for the Type-I, Type-II and Type-III Galois NLFSRs are based on the shift operation defined in Definition 2.

In [9], the Type-I Galois NLFSR is proved to be able to transformed to a Fibonacci NLFSR. However, the exact transformation process is not given in their paper. In this section, we propose transformation algorithms for the Type-I Galois NLFSRs.

Type-I-to-Fibonacci

Given an n-bit Type-I Galois NLFSR with an output function f_z, an initial state X^0 and feedback functions defined as

$$\begin{aligned} &f_{n-1} = f(x_0, \ldots, x_{n-1}) \\ &f_i = x_{i+1} \oplus f(x_0, \ldots, x_{n-1}) \quad i = \tau_1, \tau_2, \ldots, \tau_r \\ &f_i = x_{i+1} \quad \text{the rest of } i, \end{aligned} \tag{12}$$

where $0 \leq \tau_1 \leq \tau_2 \leq \ldots \leq \tau_r \leq n-1$. The transformation process is:

Step 1: For each $f_i, i = \tau_1, \tau_2, \ldots, \tau_r$, remove $f(x_0, \ldots, x_{n-1})$ and construct a compensation list $C_{f_i} = [x_{n-1-i}, x_{n-i}, \ldots, x_{n-1}, 0, \ldots, 0]$;

Step 2: Compute the combined compensation list as $C = C_{f_{\tau_1}} \oplus C_{f_{\tau_2}} \oplus \ldots \oplus C_{f_{\tau_r}}$;

Step 3: Compensate f_{n-1} by C without iteration;

Step 4: Compensate the output function f_z by C without iteration;

Step 5: The initial state $\hat{X}^0 = \{\hat{x}_0^0, \ldots, \hat{x}_{n-1}^0\}$ of transformed NLFSR is computed from $X^0 = \{x_0^0, \ldots, x_{n-1}^0\}$ and C^0 in descending order. Specifically,

$$\begin{aligned} &\hat{x}_{n-1}^0 = x_{n-1} \\ &\ldots \\ &\hat{x}_{\tau_r+1}^0 = x_{\tau_r+1}^0 \\ &\hat{x}_{\tau_r}^0 = x_{\tau_r}^0 \oplus c_{\tau_r}(\hat{X}^0) \\ &\ldots \\ &\hat{x}_0^0 = x_0^0 \oplus c_0(\hat{X}^0). \end{aligned} \tag{13}$$

The transformed NLFSR is a Fibonacci NLFSR and it is equivalent to the original Type-I Galois NLFR.

The pseudocode for the Type-I-to-Fibonacci transformation algorithm is

Algorithm 4 Type-I-to-Fibonacci

procedure TypeITF(n, $f_0, \ldots, f_{n-1}$, f_z, X^0)
  $C \leftarrow [0, \ldots, 0]$
  for $i \leftarrow [\tau_1, \ldots, \tau_r]$ **do**
    $f_i \leftarrow f_i \oplus f$ ▷ remove all the monomials in f from f_i
    $C_{f_i} \leftarrow [x_{n-1-i}, x_{n-i}, \ldots, x_{n-1}, 0, \ldots, 0]$ ▷ construct compensation list C_{f_i}
  $C \leftarrow C \oplus C_{f_i}$ ▷ combine all the compensation lists
  $\overline{f_{n-1}} \leftarrow f_{n-1}(x_0 \oplus c_0, x_1 \oplus c_1, \ldots, x_{n-1} \oplus c_{n-1})$ ▷ compensate feedback function without iteration
  $\overline{f_z} \leftarrow f_z(x_0 \oplus c_0, x_1 \oplus c_1, \ldots, x_{n-1} \oplus c_{n-1})$ ▷ compensate output function without iteration
  $\hat{X}^0 \leftarrow [0, \ldots, 0]$ ▷ calculate initial state
  for $i \leftarrow n-1, \tau_r+1$ **do**
    $\hat{x}_i^{\,0} \leftarrow x_i^0$
  $\hat{X}^0[i] \leftarrow \hat{x}_i^0$
  for $i \leftarrow \tau_r, 0$ **do**
    $c_i(\hat{X}^0) \leftarrow c_i(0, \ldots, 0, \hat{x}^0_{\tau_r+1}, \ldots, \hat{x}^0_{n-1})$
    $\hat{x}_i^0 \leftarrow x_i^0 \oplus c_i(\hat{X}^0)$
  $\hat{X}^0[i] \leftarrow \hat{x}_i^0$
return $f_0, \ldots, f_{n-2}, \overline{f_{n-1}}, \overline{f_z}, \hat{X}^0$

Appendix 2

1. Fibonacci-to-Type-II Transformation Algorithm

Given an n-bit Fibonacci NLFSR with a output function f_z and an initial state X^0, the transformation process is:

Step 1: Shift r monomials $m_0, m_1, \ldots, m_{r-1}$ from the feedback function f_{n-1} to $f_{b_1}, f_{b_2}, \ldots, f_{b_r}, 0 \leq b_0 \leq b_1 \leq \ldots \leq b_{r-1} \leq n-1$ respectively;

Step 2: The output function $\overline{f_z}$ is compensated from f_z by a compensation list C, where C is the combination of all the compensation lists $C_{m_0}, \ldots, C_{m_{r-1}}$ for each monomial.

Step 3: The initial state is computed as $\hat{x}_i^0 = x_i^0 \oplus c_i(X^0), i \in [0, n-1]$.

The pseudocode for the Fibonacci-to-Type-II transformation algorithm is

Algorithm 5 Fibonacci-to-Type-II

procedure FTTypeII(n, $f_0, \ldots, f_{n-1}$, f_z, $m_0, \ldots, m_{r-1}$, $b_0, \ldots, b_{r-1}$, X^0)
  $C \leftarrow [0, \ldots, 0]$
  for $j \leftarrow 0, r-1$ **do**
    $f_{n-1} \leftarrow f_{n-1} \oplus m_j$ ▷ shift monomial m_j from f_{n-1}
    $m_j|_{-(n-1-b_j)} \leftarrow m_j$ with $dep(m_j) - (n-1-b_j)$
    $f_{b_j} \leftarrow f_{b_j} \oplus m_j|_{-(n-1-b_j)}$ ▷ shift monomial m_j to f_{b_j}
    $C_{m_j} \leftarrow [0, \ldots, 0]$
    for $k \leftarrow b_j + 1, n-1$ **do**
      $m_j|_{-(n-k)} \leftarrow m_j$ with $dep(m_j) - (n-k)$
      $C_{m_j}[k] \leftarrow m_j|_{-(n-k)}$
    $C \leftarrow C \oplus C_{m_j}$ ▷ combine all the compensation lists
  for $i \leftarrow n-1, 0$ **do**
    if $c_i \neq 0$ **then** ▷ c_i is the i-th element in C
      $f_z \leftarrow f_z$ with x_i replaced by $x_i \oplus c_i$ ▷ compensate f_z by C iteratively
  $\overline{f_z} \leftarrow f_z$
  for $i \leftarrow 0, n-1$ **do**
    $\hat{x}_i^0 = x_i^0 \oplus c_i(X^0)$
    $\hat{X}^0[i] \leftarrow \hat{x}_i^0$ ▷ calculate initial state for the transformed NLFSR
  return $f_0, \ldots, f_{n-1}$, $\overline{f_z}$, $\hat{X}^0$

2. Type-II-to-Fibonacci Transformation Algorithm

Given an n-bit Type-II Galois NLFSR with an output function f_z and an initial value X^0, the transformation process is:

Step 1: Shift all the monomials in g_i from $f_i, i \in [b, n-2]$ to f_{n-1};

Step 2: The output function of the transformed NLFSR is constructed as $\overline{f_z} = f_z(x_0 \oplus c_0, \ldots, x_{n-1} \oplus c_{n-1})$, where $C = [c_0, \ldots, c_{n-1}]$ is the combination of all the compensation lists C_{g_i};

Step 3: The initial state is computed as $\hat{x}_i^0 = x_i^0 \oplus \overline{c_i}(X^0)$, where $\overline{c_i}$ is the monomials in c_i compensated by C iteratively.

The pseudocode for the Type-II-to-Fibonacci transformation algorithm is

Algorithm 6 Type-II-to-Fibonacci

procedure TypeIITF(n, $f_0, \ldots, f_{n-1}$, b, f_z, X^0)
  $C \leftarrow [0, \ldots, 0]$
  for $j \leftarrow 0, n-2$ **do**
    $f_j \leftarrow f_j \oplus g_j$ ▷ shift all the monomials in g_j from f_{n-1}
    $g_j|_{+(n-1-j)} \leftarrow g_j$ with $dep(g_j) + (n-1-j)$
    $f_j \leftarrow f_j \oplus g_j|_{-(n-1-b_j)}$
    $C_{g_j} \leftarrow [0, \ldots, 0]$
    for $k \leftarrow b, n-2$ **do**
      $g_j|_{+(n-k)} \leftarrow g_j$ with $dep(g_j) + (n-k)$
      $C_{g_j}[k] \leftarrow g_j|_{-(n-k)}$ ▷ construct compensation list C_{g_j}
    $C \leftarrow C \oplus C_{g_j}$ ▷ combine all the compensation lists
  for $i \leftarrow 0, n-1$ **do** ▷ we start the loop from compensating x_0 to x_{n-1}
    if $c_i \neq 0$ **then**
      replace x_i by $x_i \oplus c_i$ in f_z ▷ no stage is iteratively compensated
  $\overline{f_z} \leftarrow f_z$
  for $i \leftarrow 0, n-1$ **do**
    $\overline{c_i} \leftarrow c_i$
    for $j \leftarrow n-1, 0$ **do**
      if $c_j \neq 0$ **then**
        $\overline{c_i} \leftarrow \overline{c_i}$ with x_j replaced by $x_j \oplus c_j$ ▷ get compensated $\overline{c_i}$
    $\hat{x}_i^0 = x_i^0 \oplus \overline{c_i}(X^0)$ ▷ calculate initial state
    $\hat{X}^0[i] \leftarrow \hat{x}_i^0$
  return $f_0, \ldots, f_{n-1}$, $\overline{f_z}$, $\hat{X}^0$

Appendix 3

1. Fibonacci-to-Type-III Transformation Algorithm

Given an n-bit Fibonacci NLFSR with an output function f_z and an initial value X^0, the transformation process is:

Step 1: Shift r monomials $m_0, m_1, \ldots, m_{r-1}$ from f_{n-1} to feedback functions f_{b_0}, $\ldots, f_{b_{r-1}}, 0 \leq b_0 < b_1 < \ldots < b_{r-1} \leq n-2$ respectively;

Step 2: The feedback functions and output function are compensated by the compensation list C;

Step 3: The initial state is $\hat{X}^0 = \{\hat{x}_i^0 = x_i^0 \oplus c_i(X^0), i \in [0, n-1]\}$.

The pseudocode for Fibonacci-to-Type-III algorithm is in Algorithm 7.

Algorithm 7 Fibonacci-to-Type-III

procedure FTTypeIII(n, $f_0, \ldots, f_{n-1}$, f_z, $m_0, \ldots, m_{r-1}$, $b_0, \ldots, b_{r-1}$, X^0)
  $C \leftarrow [0, \ldots, 0]$
  for $j \leftarrow 0, r-1$ **do**
    $f_{n-1} \leftarrow f_{n-1} \oplus m_j$
    $m_j|_{-(n-1-b_j)} \leftarrow m_j$ with $dep(m_j) - (n-1-b_j)$
    $f_{b_j} \leftarrow f_{b_j} \oplus m_j|_{-(n-1-b_j)}$
    $C_{m_j} \leftarrow [0, \ldots, 0]$
    for $k \leftarrow b_j + 1, n-1$ **do**
      $m_j|_{-(n-k)} \leftarrow m_j$ with $dep(m_j) - (n-k)$
      $C_{m_j}[k] \leftarrow m_j|_{-(n-k)}$
    $C \leftarrow C \oplus C_{m_j}$
  for $i \leftarrow n-1, 0$ **do**
    if $c_i \neq 0$ **then**
      $f_z \leftarrow f_z$ with x_i replaced by $x_i \oplus c_i$
      for $j \leftarrow 0, n-1$ **do** ▷ compensate all the feedback functions
        $f_{b_j} \leftarrow f_{b_j}$ with x_i replaced by $x_i \oplus c_i$
  $\overline{f_z} \leftarrow f_z$
  for $i \leftarrow 0, n-1$ **do**
    $\hat{x}_i^0 = x_i^0 \oplus c_i(X^0)$
    $\hat{X}^0[i] \leftarrow \hat{x}_i^0$
  return $f_0, \ldots, f_{n-1}$, $\overline{f_z}$, $\hat{X}^0$

2. Type-III-to-Fibonacci Transformation Algorithm

Given an n-bit Type-IV Galois NLFSR with an output function f_z and an initial value X^0, all the monomials in g_i are shifted from $f_i, i \in [0, n-2]$ to f_{n-1}, the Galois NLFSR is transformed to a Fibonacci NLFSR by following steps:

Step 1: Let the combined compensation list $C = [0, \ldots, 0]$;

Step 2: For each $g_i, i \in [0, n-2]$, starts from $i = 0$, remove it from f_i and construct a compensation list $C_i = [0, \ldots, 0, g_i, g_i|_{+1}, \ldots, g_i|_{+(n-i)}]$ and compute $C = C \oplus C_i$;

Step 3: Compensation: only use $C[i+1]$ to compensate the tap x_{i+1} in all the feedback functions, which means x_{i+1} is replaced by $x_{i+1} \oplus C[i+1]$;

Step 4: If $i \neq n-2$, then set $i = i+1$ and go back to Step 2, otherwise, go to Step 5;

Step 5: The final combined compensation list is C. Xor all the shifted monomials $g_i|_{+(n-i)}, i \in [0, n-2]$ to the feedback function of bit x_{n-1} to get the final feedback functions for the transformed NLFSR. The output function f'_z is constructed by replacing $x_i, i \in [0, n-1]$ in f_z by $x_i \oplus c_i$, no iteration is needed in this compensation process;

Step 6: The initial value $\hat{X}^0 = \{\hat{x}^0_0, \ldots, \hat{x}_{n-1}\}$ is computed by compensating $X^0 = \{x^0_0, \ldots, x^0_{n-1}\}$ by C^0 iteratively starting from $i = n-1$ to $i = 0$.

The pseudocode for the Type-II-to-Fibonacci transformation algorithm is

Algorithm 8 Type-III-to-Fibonacci

procedure TypeIIITF(n, $f_0, \ldots, f_{n-1}$, f_z, X^0)
  $C_{g_i} \leftarrow [0, \ldots, 0]$
  for $i \leftarrow 0, n-2$ **do**
    $f_i \leftarrow f_i \oplus g_i$ ▷ shift monomials in g_i from f_i
    $C_{g_i} \leftarrow [0, \ldots, 0]$
    for $j \leftarrow i+1, n-1$ **do**
      $g_i|_{+(j-i-1)} \leftarrow g_i$ with $dep(g_i) + (j-i-1)$
      $C_{g_i}[j] \leftarrow g_i|_{+(j-i-1)}$ ▷ construct compensation list
    $C \leftarrow C \oplus C_{g_i}$ ▷ combine all the compensation lists
    for $j \leftarrow 0, n-1$ **do**
      $f_j \leftarrow f_j$ with x_{i+1} replaced by $x_{i+1} \oplus c_{i+1}$ ▷ compensate only x_{i+1}
    $f_z \leftarrow f_z$ with x_{i+1} replaced by $x_{i+1} \oplus c_{i+1}$
  $\overline{f_z} \leftarrow f_z$
  for $i \leftarrow 0, n-2$ **do**
    $f_{n-1} \leftarrow f_{n-1} \oplus g_i|_{+(n-i)}$ ▷ shift monomials to f_{n-1}
  for $i \leftarrow 0, n-1$ **do**
    $\overline{c_i} \leftarrow c_i$
    for $j \leftarrow n-1, 0$ **do**
      if $C[j] \neq 0$ **then**
        $\overline{c_i} \leftarrow \overline{c_i}$ with x_j replaced by $x_j \oplus c_j$ ▷ get compensated $\overline{c_i}$
    $\hat{x}^0_i = x^0_i \oplus \overline{c_i}(X^0)$
    $\hat{X}^0[i] \leftarrow \hat{x}^0_i$
  return $f_0, \ldots, f_{n-1}$, $\overline{f_z}$, $\hat{X}^0$