A Survey and New Results on the Decomposition of an NFSR into a Cascade Connection of Two Smaller NFSRs*

Tian Tian, Jia-Min Zhang, Chen-Dong Ye, and Wen-Feng Qi†‡

February 10, 2018

Abstract

Nonlinear feedback shift registers (NFSRs) are an important building block for stream ciphers. Given a cascade connection of two NFSRs, say NFSR(f, g), it has been known for decades how to solve the characteristic function of the NFSR which is equivalent to NFSR(f, g). However, the converse problem of decomposing an NFSR into a cascade connection of two smaller NFSRs is not completely solved, and only a special case has been studied recently. In this paper, a complete and feasible solution to the problem is given.

Keywords: Stream ciphers, nonlinear feedback shift registers, cascade connection of NFSRs, decomposition.

*This work was supported by the National Natural Science Foundation of China (Grant 61672533, 61521003).

†Tian Tian, Jia-Min Zhang, Chen-Dong Ye, and Wen-Feng Qi are with National Digital Switching System Engineering & Technological Research Center, P.O. Box 407, 62 Kexue Road, Zhengzhou, 450001, China.(e-mail: tiantian_d@126.com, ye_chendong@126.com, jiamin.zhang.edu@gmail.com, and wenfeng.qi@263.net).

‡Corresponding author: Tian Tian

1 Introduction

Linear feedback shift registers (LFSRs) are the most popular building block used to design stream ciphers, for they have very good statistical properties, efficient implementations and well studied algebraic structures. Yet over the years, stream ciphers based on LFSRs have been found to be susceptible to algebraic attacks and correlation attacks. Therefore, many recently proposed stream ciphers adopt nonlinear sequence generators. In particular nonlinear feedback shift registers (NFSRs) is an important type of nonlinear sequence generators with more than 50 years of research. Grain and Trivium, two eSTREAM hardware-oriented finalists, use NFSRs as a main building block, see [1, 2]. Now Trivium has been specified as an International Standard under ISO/IEC 29192-3:2012. Besides, NFSRs are also used for block cipher design, say KATAN [3]. In this paper we are concerned with cascade connection decomposition of NFSRs.

Cascade connections of two NFSRs were firstly studied in [4], which was proposed as a generalization of the case of LFSRs. Let $f_1(x)$ and $f_2(x)$ be two polynomials over $\mathbb{F}_2$, the finite field of two elements. It is known that a sequential circuit made up from a cascade connection of the LFSR with the characteristic polynomial $f_1(x)$ into the LFSR with the characteristic polynomial $f_2(x)$ outputs the same family of sequences as the LFSR with the characteristic polynomial $f_1(x)f_2(x)$ [4]. Thus a product LFSR (an LFSR with a composite characteristic polynomial) can be interpreted as a cascade connection of its factors. In [4] the author demonstrated similar equivalence for the nonlinear case by introducing an order increasing multiplication to Boolean functions which is denoted by "$*$" in the following paper to distinguish from traditional multiplication "$\cdot$" (see Section 2). It was shown in [4] that a cascade connection of the NFSR F_1 with the characteristic function $f_1(x_0, x_1, \ldots, x_n)$ into the NFSR F_2 with the characteristic function $f_2(x_0, x_1, \ldots, x_m)$ outputs the same family of sequences as the NFSR F_3 with the characteristic function $f_1 * f_2$. However, the converse problem of decomposing an NFSR into a cascade connection of two smaller NFSRs was not covered in [4].

Such decomposition problem was firstly studied in [5] which only considered a special case that is decomposing an NFSR into the cascade connection of an NFSR into an *LFSR*. Later, the authors in [6] improved the results of [5] in many aspects. In this paper, we provide a complete and feasible solution to the problem. First, we review previous results given in [5] and [6] as well as give a new insight on the special decomposition case. Second, we solve another special case of decomposing an NFSR into the cascade connection of an *LFSR* into an NFSR. Third, we discuss the problem of decomposing an NFSR into the cascade connection of an NFSR into an NFSR where no NFSR is degenerated to an LFSR, which is the most general case. We prove an important algebraic property of a cascade connection of two NFSRs, and propose a decomposition algorithm based on it. The complexity of the proposed algorithm for the general case is closely related to the Algebraic Normal Form (ANF) of the characteristic function. Generally speaking, the characteristic functions with less terms and lower degree and smaller order (see Section 2 for the definition of order) are easier to decompose. We think this is reasonable for an algorithm solving problems concerning NFSRs. An extensive experiments show that the proposed algorithm is feasible in practice, e.g., decompose an 80-stage NFSR with characteristic function including 100 terms and of degree 4 in about 10 minutes.

The paper is organized as follows. Section 2 presents an introduction to Boolean functions and NFSRs. Section 3 discusses the uniqueness of left $*$-factor and also some basic properties of $*$-product. Section 4 completely solves the decomposition of an NFSR into a cascade connection of an *LFSR* into an NFSR. Section 5 is largely devoted to the survey of previous results on the decomposition of an NFSR into a cascade connection of an NFSR into an *LFSR*, and also includes a small new result on such decomposition. Section 6 discusses general decomposition case. Finally, conclusions are drawn in Section 7.

Throughout the paper, the set $\{0, 1, 2, \ldots\}$ of nonnegative integers is denoted by $\mathbb{N}$, the set $\{1, 2, \ldots\}$ of positive integers is denoted by $\mathbb{N}^*$, and the symbol $\oplus$ denotes addition modulo 2. We use the abbreviation w.r.t. for the phrase "with respect to".

2 Preliminaries

In this section, we briefly review Boolean functions and nonlinear feedback shift registers respectively. We remark that a nonlinear feedback shift register can be described by a Boolean function called characteristic function.

2.1 Boolean functions

Let $n \in \mathbb{N}^*$. An n-variable Boolean function $f(x_0, x_1, \ldots, x_{n-1})$ is a function from $\mathbb{F}_2^n$ into $\mathbb{F}_2$ and the set of all n-variable Boolean functions is denoted by $\mathcal{B}_n$. It is known that an n-variable Boolean function $f(x_0, x_1, \ldots, x_{n-1})$ can be uniquely represented as a multivariate polynomial of the form:

$$f(x_0, x_1, \ldots, x_{n-1}) = \bigoplus_{\alpha=(\alpha_0,\alpha_1,\ldots,\alpha_{n-1})\in\{0,1\}^n} u_\alpha \cdot \left(\prod_{j=0}^{n-1} x_j^{\alpha_j} \right),$$

where $u_\alpha \in \mathbb{F}_2$, which is called the *algebraic normal form* (ANF) of f. The *algebraic degree* of f, denoted by $\deg(f)$, is the global degree of the ANF of f. If $\deg(f) = 1$ and $f(0, 0, \ldots, 0) = 0$, then we say f is *linear*. If $\deg(f) \geq 1$, then the highest subscript i for which x_i occurs in the ANF of f is called the *order* of f and denoted by $\mathrm{ord}(f)$.

A product of the form $x_0^{\alpha_0} x_1^{\alpha_1} \cdots x_{n-1}^{\alpha_{n-1}} \in \mathcal{B}_n$ with $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \in \{0,1\}^n$ is called a *term*; in particular, $1 = x_0^0 x_1^0 \cdots x_{n-1}^0$ is a term. Let us denote the set of all terms in $\mathcal{B}_n$ by $T(x_0, x_1, \ldots, x_{n-1})$. The term order, *inverse lexicographical order* $\preceq$, is used throughout the paper, which is defined by

$$x_0^{\alpha_0} x_1^{\alpha_1} \cdots x_{n-1}^{\alpha_{n-1}} \preceq x_0^{\beta_0} x_1^{\beta_1} \cdots x_{n-1}^{\beta_{n-1}}$$

if and only if

$$\alpha_0 + \alpha_1 \cdot 2 + \cdots + \alpha_{n-1} \cdot 2^{n-1} \leq \beta_0 + \beta_1 \cdot 2 + \cdots + \beta_{n-1} \cdot 2^{n-1}$$

holds. Moreover, for $t, s \in T(x_0, x_1, \ldots, x_{n-1})$, we write $t \prec s$ if $t \preceq s$ and $t \neq s$. In

particular, we have that

$$1 \prec x_0 \prec x_1 \prec \cdots \prec x_{n-1}.$$

For $f \in \mathcal{B}_n$ we denote the head term of f with respect to the term order by $\mathrm{HT}(f)$ and denote the set of all terms occurring in the ANF of f by $T(f)$. If all terms of f have the same degree, then we say f is *homogenous*. Otherwise, f can be uniquely written as a finite sum of homogenous Boolean functions: $f = \bigoplus_{d=0}^{\deg(f)} f_{[d]}$, where $f_{[d]}$ is the summation of all terms of f that have degree d. Moreover, we denote the summation of all nonlinear terms of f by $NL(f)$, i.e., $NL(f) = \bigoplus_{d=2}^{\deg(f)} f_{[d]}$.

Let $m \in \mathbb{N}^*$. For $f \in \mathcal{B}_n$ and $g \in \mathcal{B}_m$, let us denote

$$f * g = f(g(x_0, \ldots, x_{m-1}), g(x_1, \ldots, x_m), \ldots, g(x_{n-1}, \ldots, x_{n+m-2})), \tag{1}$$

which is an $(n + m - 1)$-variable Boolean function. Note that the operation $*$ is not commutative, that is, $f * g$ and $g * f$ are not the same in general. If $h = f * g$, then we say f is a *left* $*$-factor of h and g is a *right* $*$-factor of h. Clearly for all $h \in \mathcal{B}_n$, we have that $h = h * x_0 = x_0 * h$, and so h and x_0 are called *trivial* $*$-factors of h.

The following properties of the operation $*$ are directly deduced from its definition (1), which will be frequently used in the following paper.

Proposition 1 *Let $f, g, q \in \mathcal{B}_n$. Then*

(i) $(f \cdot q) * g = (f * g) \cdot (q * g)$;

(ii) $f * g = \bigoplus_{t \in T(f)} (t * g)$;

(iv) $f * g = \bigoplus_{t \in T(f)} \bigoplus_{s \in T(g)} t * s$ *if f is linear.*

In the next subsection, we will give the cryptographic background for this $*$-product of Boolean functions.

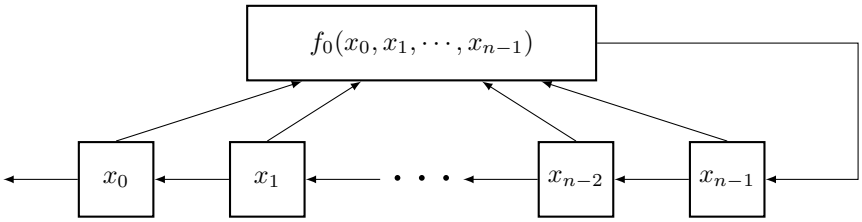

Figure 1: An n-stage NFSR

Finally, for a linear Boolean function $f = c_0x_0 \oplus c_1x_1 \oplus \cdots \oplus c_{n-1}x_{n-1}$, define

$$\phi(f) = c_0 \oplus c_1x \oplus \cdots \oplus c_{n-1}x^{n-1} \in \mathbb{F}_2[x]. \tag{2}$$

The function ϕ maps a linear Boolean function to a univariate polynomial over $\mathbb{F}_2$, which is a one-to-one correspondence. It can be seen that $\phi(f * g) = \phi(f)\phi(g)$ holds for two linear Boolean functions f and g. Thus, for simplicity, we may directly treat a linear Boolean function as a univariate polynomial over $\mathbb{F}_2$ or conversely treat a univariate polynomial over $\mathbb{F}_2$ as a linear Boolean function omitting the symbol ϕ and ϕ^{-1}.

2.2 Nonlinear feedback shift registers

Let $n \in \mathbb{N}^*$. A diagram of an n-stage NFSR with characteristic function

$$f(x_0, x_1, \ldots, x_n) = f_0(x_0, x_1, \ldots, x_{n-1}) \oplus x_n \in \mathcal{B}_{n+1}$$

is given in Figure 1, denoted by NFSR(f), where $f_0(x_0, x_1, \ldots, x_{n-1})$ is usually called the feedback function of the NFSR in the literature.

An output sequence $\underline{s} = (s_t)_{t\geq 0}$ of the NFSR(f) is a binary sequence satisfying the following recurrence relation

$$s_{t+n} = f_0(s_t, s_{t+1}, \ldots, s_{t+n-1}), \text{ for } t \geq 0.$$

In particular, if $f(x_0, x_1, \ldots, x_n)$ is linear, then the NFSR(f) is also known as an LFSR with characteristic polynomial $\phi(f)$. The set of all 2^n sequences generated by the NFSR(f)

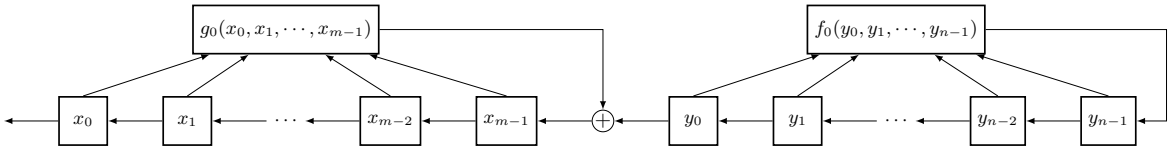

Figure 2: The cascade connection of the NFSR(f) into the NFSR(g)

is denoted by $G(f)$. It is well known that all sequences in $G(f)$ are (strictly) periodic if and only if $f(x_0, x_1, \ldots, x_n)$ is nonsingular, namely $f(x_0, x_1, \ldots, x_n) = x_0 \oplus f_1(x_1, x_2, \ldots, x_{n-1}) \oplus x_n$, see [7, Chapter VI]. For convenience, let us denote

$$\mathcal{C} = \{f \mid f(x_0, x_1, \ldots, x_r) = x_0 \oplus f_1(x_1, x_2, \ldots, x_{r-1}) \oplus x_r \in \mathcal{B}_{r+1}, r \in \mathbb{N}^*\},$$

the set of all nonsingular characteristic functions. We further denote

$$\mathcal{C}^* = \{f(x_0, x_1, \ldots, x_r) \in \mathcal{C} \mid f(0, 0, \ldots, 0) = 0 \},$$

the set of nonsingular characteristic functions which outputs the all-zero sequence.

Let $m \in \mathbb{N}^*$ and $g(x_0, x_1 \ldots, x_m) = g_0(x_0, x_1, \ldots, x_{m-1}) \oplus x_m \in \mathcal{B}_{m+1}$. The Galois NFSR shown in Figure 2 is called the *cascade connection* of the NFSR(f) into the NFSR(g), denoted by NFSR(f, g), where to distinguish the registers of two NFSRs, the registers belonging to the NFSR(f) are labeled $y_0, y_1, \ldots, y_{n-1}$. An output sequence of the register labeled x_0 is called an output sequence of the NFSR(f, g) and the set of all output sequences of the NFSR(f, g) is denoted by $G(f, g)$. It was early known that the NFSR(f, g) is equivalent to the NFSR(h) where $h = f * g$, namely $G(f, g) = G(h)$, see [4] and [8].

In the following, **we specifically discuss how to decompose an NFSR in $\mathcal{C}^*$ into a cascade connection of two NFSRs in $\mathcal{C}^*$**. First if an NFSR in $\mathcal{C}^*$ can be decomposed into two NFSRs in $\mathcal{C}$, then it also can be decomposed into two NFSRs in $\mathcal{C}^*$ (Please see Appendix A for the proof). Second, if $h \oplus 1 = f * g$ where $h \in \mathcal{C}^*$, then $h = (f \oplus 1) * g$. Thus, it suffices to discuss decomposition within $\mathcal{C}^*$.

Finally, since a cascade connection of NFSRs is algebraically represented by the $*$-product operation of Boolean functions, in the following we in essence discuss finding the left and right $*$-product factors of characteristic functions.

3 The Uniqueness of Left $*$-Factor

In this section, we shall show some basic properties of the $*$-product operation which will be frequently used later.

Lemma 2 *Let* $m, n \in \mathbb{N}^*$, $g(x_0, \ldots, x_m) = g_0(x_0, \ldots, x_{m-1}) \oplus x_m \in \mathcal{B}_{m+1}$, *and* $t = x_{i_1} x_{i_2} \cdots x_{i_k} \in T(x_0, \ldots, x_n)$, *where* $k \geq 1$ *and* $i_1 < i_2 < \cdots < i_k$. *Then*

(i) $\mathrm{HT}(t * g) = t * x_m = \prod_{j=1}^{k} x_{m+i_j}$;

(ii) $\deg(t * g) \geq \deg(g) + \deg(t) - 1$. *In particular, the equality holds for* $k = 1$.

Proof. (i) Since

$$t * g = \prod_{j=1}^{k} \left(g_0(x_{i_j}, \ldots, x_{m-1+i_j}) \oplus x_{m+i_j} \right), \tag{3}$$

it follows that

$$\mathrm{HT}(t * g) = \prod_{j=1}^{k} x_{m+i_j}.$$

(ii) The assertion is trivially true for $k = 1$. We suppose $k > 1$. By (3), $t * g$ can be written

$$t * g = (g_0(x_{i_1}, \ldots, x_{m-1+i_1}) \oplus x_{m+i_1}) \cdot \left(\prod_{j=2}^{k} x_{m+i_j} \right) \oplus u(x_{i_1}, \ldots, x_{m+i_k}), \tag{4}$$

where

$$\prod_{j=2}^{k} x_{m+i_j} \nmid s \text{ for all } s \in T(u).$$

Since for $j = 2, 3, \ldots, k$,

$$m + i_j > m + i_1 = \mathrm{ord}(g_0(x_{i_1}, \ldots, x_{m-1+i_1}) \oplus x_{m+i_1}),$$

it follows from (4) that

$$s \cdot \prod_{j=2}^{k} x_{m+i_j} \in T(t * g)$$

for all $s \in T\left(g_0(x_{i_1}, \ldots, x_{m-1+i_1}) \oplus x_{m+i_1}\right)$. Let

$$s^* \in T\left(g_0(x_{i_1}, \ldots, x_{m-1+i_1}) \oplus x_{m+i_1}\right)$$

such that $\deg(s^*) = \deg(g)$. Then we have that

$$s^* \cdot \prod_{j=2}^{k} x_{m+i_j} \in T(t * g), \tag{5}$$

and

$$\deg(s^* \cdot \prod_{j=2}^{k} x_{m+i_j}) = \deg(s^*) + k - 1 = \deg(g) + \deg(t) - 1. \tag{6}$$

Thus the assertion follows from (5) and (6) for $k > 1$. ■

Remark 3 *If g is not of the form described in Lemma 2, then the results may not hold. For instance, $(x_3x_4) * (x_1x_2 \oplus x_2) = 0$.*

Corollary 4 *Let $m \in \mathbb{N}^*$ and $g(x_0, \ldots, x_m) = g_0(x_0, \ldots, x_{m-1}) \oplus x_m \in \mathcal{B}_{m+1}$. Then for any Boolean function f which is not a constant, $f * g \neq 0$ and $\mathrm{HT}(f * g) = \mathrm{HT}(f) * x_m$.*

Proof. The assertion follows from Lemma 2 (i) and the fact $f * g = \sum_{t \in T(f)} t * g$. ■

If $g, f_1, f_2 \in \mathcal{C}^*$ such that $f_1 * g = f_2 * g$, then it follows from Corollary 4 that $f_1 = f_2$ since $(f_1 \oplus f_2) * g = 0$. Thus we have the following uniqueness.

Corollary 5 *Let $h, g \in \mathcal{C}^*$. If g is a right $*$-factor of h , then there exists a unique Boolean function $f \in \mathcal{C}^*$ such that $h = f * g$.*

4 Decompose A Boolean Function h into $h = l * f$

In this section, we shall prove that the linear left $*$-factor of the maximal order for a Boolean function is unique and show how to obtain it. Trivially, the results of this section can be directly used to decompose an NFSR into a cascade connection of an LFSR into an NFSR. Besides, the results of this section will be used in Section 6.

Lemma 6 *Let h, g be two Boolean functions which are not constants, and let l be a linear Boolean function. Then $h = l * g$ if and only if $h_{[i]} = l * g_{[i]}$ for $1 \leq i \leq \deg(h)$.*

Proof. Suppose $h = l * g$. Since l is linear, it is easily seen that

$$h = \bigoplus_{i=1}^{\deg(g)} l * g_{[i]}.$$

It follows from Lemma 2 (ii) that $l * g_{[i]}$ is a homogenous Boolean function of degree i for $1 \leq i \leq \deg(g)$. Thus, $h_{[i]} = l * g_{[i]}$ for $1 \leq i \leq \deg(h)$. The converse is trivially true. ■

It follows from Lemma 6 that the key problem is decomposing homogenous Boolean functions. Firstly, we consider a homogenous Boolean function obtained by $l * t$ where l is a linear Boolean function and t is a term.

Lemma 7 *Let h be a homogenous Boolean function of degree $d \geq 1$. Then $h = l * s$ for some linear Boolean function l and some term s if and only if any two terms $t = x_{i_1} x_{i_2} \cdots x_{i_d}$ and $t' = x_{j_1} x_{j_2} \cdots x_{j_d}$ in $T(h)$ satisfy that*

$$(i_2 - i_1, i_3 - i_1, \ldots, i_d - i_1) = (j_2 - j_1, j_3 - j_1, \ldots, j_d - j_1).$$

Proof. ($\Leftarrow$) Let $t = x_{i_1} x_{i_2} \cdots x_{i_d}$ be a term of h and let

$$(i_2 - i_1, i_3 - i_1, \ldots, i_d - i_1) = (\delta_1, \delta_2, \ldots, \delta_{d-1}).$$

Then it is clear that we can write

$$t = x_{i_1} x_{i_1+\delta_1} \cdots x_{i_1+\delta_{d-1}}.$$

Moreover, according to the hypothesis, we can write

$$h = \sum_{k=1}^{|T(h)|} x_{i_k} x_{i_k+\delta_1} \cdots x_{i_k+\delta_{d-1}}. \tag{7}$$

Without loss of generality, we assume that $i_1 < i_2 < \cdots < i_{|T(h)|}$. Then it follows from (7) that

$$\begin{aligned} h &= \sum_{k=1}^{|T(h)|} x_{i_1+(i_k-i_1)} x_{i_1+\delta_1+(i_k-i_1)} \cdots x_{i_1+\delta_{d-1}+(i_k-i_1)} \\ &= (\sum_{k=1}^{|T(h)|} x_{i_k-i_1}) * (x_{i_1} x_{i_1+\delta_1} \cdots x_{i_1+\delta_{d-1}}) \\ &= (\sum_{k=1}^{|T(h)|} x_{i_k-i_1}) * t. \end{aligned}$$

Thus $l = \sum_{k=1}^{|T(h)|} x_{i_k-i_1}$ is the desired linear Boolean function.

($\Rightarrow$) The converse is trivially true. ■

Remark 8 *In the following section, for a term $t = x_{i_1} x_{i_2} \cdots x_{i_d}$ we call $(i_2 - i_1, i_3 - i_1, \ldots, i_d - i_1)$ the index distance tuple of t.*

Secondly, we consider general homogenous Boolean functions. By Lemma 7, we know that if all the terms of a homogenous Boolean function h are classified by their index distance tuple and combine the terms with the same index distance tuple by $*$-product of the form $l * t$ where l is a linear Boolean function and t is a term, then the function h can be **uniquely** represented as the form

$$h = l_1 * t_1 \oplus l_2 * t_2 \oplus \cdots \oplus l_k * t_k$$

where $l_1, l_2, \ldots, l_k$ are linear Boolean functions and $t_1, t_2, \ldots, t_k$ are terms with pairwise distinct index distance tuple. Based on this observation, we immediately get the following result on homogenous Boolean functions.

Lemma 9 *Let h be a homogenous Boolean function of degree $d \geq 1$ and*

$$h = l_1 * t_1 \oplus l_2 * t_2 \oplus \cdots \oplus l_k * t_k \tag{8}$$

where $l_1, l_2, \ldots, l_k$ are linear Boolean functions and $t_1, t_2, \ldots, t_k$ are terms with pairwise distinct index distance tuple. Then $h = l * g$ for some Boolean functions l and g where l is a linear function if and only if $\phi(l)$ is a divisor of $\gcd(\phi(l_1), \phi(l_2), \ldots, \phi(l_k))$.

Proof. ($\Leftarrow$) For $i = 1, 2, \ldots, k$, let $\phi(l_i) = \phi(l)\phi(f_i)$, where f_i is a linear Boolean function. Since

$$\phi(l * f_i) = \phi(l)\phi(f_i), i = 1, 2, \ldots, k,$$

it follows that $l_i = l * f_i, i = 1, 2, \ldots, k$, and so

$$\begin{aligned} h &= (l * f_1) * t_1 \oplus (l * f_2) * t_2 \oplus \cdots \oplus (l * f_k) * t_k \\ &= l * (f_1 * t_1 \oplus f_2 * t_2 \oplus \cdots \oplus f_k * t_k). \end{aligned}$$

This shows the result holds.

($\Rightarrow$) Let us write

$$g = f_1 * s_1 \oplus f_2 * s_2 \oplus \cdots \oplus f_r * s_r$$

where $f_1, f_2, \ldots, f_r$ are linear Boolean functions and $s_1, s_2, \ldots, s_r$ are terms with pairwise distinct index distance. Then

$$\begin{aligned} h &= l * g \\ &= l * (f_1 * s_1 \oplus f_2 * s_2 \oplus \cdots \oplus f_r * s_r) \\ &= (l * f_1) * s_1 \oplus (l * f_2) * s_2 \oplus \cdots \oplus (l * f_r) * s_r \end{aligned}$$

Comparing this with (8), we know that $k = r$ and $l_i = l * f_i$ for $i = 1, 2, \ldots, k$. Thus, it immediately follows that $\phi(l)$ is a divisor of $\gcd(\phi(l_1), \phi(l_2), \ldots, \phi(l_k))$. ■

Remark 10 *For a homogenous Boolean function h described in Lemma 9, it can be seen that* $\gcd(\phi(l_1), \phi(l_2), \ldots, \phi(l_k))$ *is the unique linear left $*$-factor whose order is the largest, and so we denote it by $L(h)$.*

Finally, with these discussions on homogenous Boolean functions, the following result on general Boolean functions is an immediate consequence of lemmas 6 and 9.

Theorem 11 *Let $h \in C^*$ of degree $d > 1$ and let l be a linear Boolean function. Then $h = l * g$ for some Boolean function g if and only if $\phi(l)$ is a divisor of*

$$\gcd(L(h_{[1]}), L(h_{[2]}), \ldots, L(h_{[d]})).$$

5 Decomposition into a Cascade Connection of an NFSR into an LFSR

In this section, we discuss theories and algorithms about decomposing an NFSR into a cascade connection of an NFSR into an LFSR. This problem was extensively studied in [5] and [6]. In Subsection 5.1, we review the previous methods and results. In Subsection 5.2, we give a new candidate linear function to reduce the previous verification range.

5.1 Necessary and sufficient conditions on the decomposition

A first necessary and sufficient condition for decomposition into a cascade connection of an NFSR into an LFSR is proved in [6].

Theorem 12 *Let $h(x_0, x_1, \ldots, x_n) \in C^*$ and $l(x_0, x_1, \ldots, x_m)$ be a linear Boolean function. Then $h = g * l$ for some function g if and only if $\{(a_0, a_1, \ldots, a_n) | \underline{a} = (a_0, a_1, \ldots) \in G(l)\}$ is included in*

$$\{(e_0, e_1, \ldots, e_n) | h(x_0, x_1, \ldots, x_n) = h(x_0 \oplus e_0, x_1 \oplus e_1, \ldots, x_n \oplus e_n)\}.$$

The following corollary immediately follows from the above theorem, which presents the necessary and sufficient condition from the aspect of sequences.

Corollary 13 *Let $h \in C^*$ and l be a linear Boolean function. Then $h = g * l$ for some function g if and only if*

$$G(h) = G(h) \oplus G(l) = \{\underline{a} \oplus \underline{b} | \underline{a} \in G(h), \underline{b} \in G(l)\}.$$

Corollary 13 reveals a kind of closure property of exclusive ors of sequences generated by h, which is very rare for NFSR sequences. Moreover, it implies the following uniqueness of linear right $*$-factor.

Corollary 14 *For every Boolean function $h \in C^*$, there exists a unique linear right $*$-factor l whose order is the largest. Furthermore, all the other linear right $*$-factor of h is a factor of l.*

Unlike Theorem 11, both Theorem 12 and Corollary 13 do not tell us how to obtain the unique largest linear right $*$-factor of a function h from the ANF of h. Until now, all known algorithms only can extract a *multiple* of the largest linear right $*$-factor from the ANF of h, and so a verification step is always needed. Thus, to decompose a function $h \in C^*$ into the form $h = g * l$ where l is a linear Boolean function, one can obey the following four steps:

1. Find a candidate linear function m which is a multiple of the largest linear right $*$-factor of h.
2. Factor $\phi(m)$ in $\mathbb{F}_2[x]$.
3. Verify which factor of $\phi(m)$ is the linear right $*$-factor of h with the largest degree.
4. For the linear function l that passes through the verification of the last step, find the function g such that $h = g * l$.

To find a candidate linear function m which is a multiple of the largest linear right $*$-factor of h is the most important step. Clearly, it is expected that the candidate linear function m is close to the real largest linear right $*$-factor of h. Thus we want to find a candidate linear function m whose order (or equivalently $\deg(\phi(m))$) is as small as possible. In [6], the authors gave the following candidate linear function m which is always a factor of that given in [5].

Theorem 15 *Let $h(x_0, x_1, \ldots, x_n) \in C^*$ of degree $d > 1$. Suppose $h_{[d]}$ can be written in k ways as following*

$$h_{[d]} = t_1 \cdot l_1 \oplus h_1 = t_2 \cdot l_2 \oplus h_2 = \ldots = t_k \cdot l_k \oplus h_k,$$

where t_i is a term of degree $t-1$, l_i is a linear Boolean function, h_i is a Boolean function of degree d such that each term in h_i is not divisible by t_i, for $1 \leq i \leq k$. If l is a linear right $*$-factor of h, then $\phi(l)$ divides $\gcd(\phi(l_1), \ldots, \phi(l_k))$ in $\mathbb{F}_2[x]$.

Though we can not prove that the function $\gcd(\phi(l_1), \ldots, \phi(l_k))$ given by Theorem 15 is just the largest linear right $*$-factor of h, it is very likely to be the one.

Example 16 *Let*

$$\begin{aligned} h \;=\; & x_7 \oplus x_4x_5x_6 \oplus x_2x_5x_6 \oplus x_1x_5x_6 \oplus x_3x_4x_6 \oplus x_2x_4x_6 \oplus x_4x_6 \oplus x_2x_3x_6 \\ & \oplus x_1x_3x_6 \oplus x_1x_2x_6 \oplus x_1x_6 \oplus x_3x_4x_5 \oplus x_2x_4x_5 \oplus x_1x_4x_5 \oplus x_4x_5 \oplus x_2x_3x_5 \\ & \oplus x_1x_3x_5 \oplus x_5 \oplus x_1x_3x_4 \oplus x_3x_4 \oplus x_1x_2x_4 \oplus x_2x_4 \oplus x_1x_4 \oplus x_4 \oplus x_1x_2x_3 \\ & \oplus x_2x_3 \oplus x_3 \oplus x_2 \oplus x_0 \end{aligned}$$

be a characteristic function of an NFSR. It can be seen that $\deg(h) = 3$ and $h_{[3]}$ can be written in the following ways

$$\begin{aligned} h_{[3]} \;=\; & x_4x_5x_6 \oplus x_2x_5x_6 \oplus x_1x_5x_6 \oplus x_3x_4x_6 \oplus x_2x_4x_6 \oplus x_2x_3x_6 \oplus x_1x_3x_6 \oplus x_1x_2x_6 \\ & \oplus x_3x_4x_5 \oplus x_2x_4x_5 \oplus x_1x_4x_5 \oplus x_2x_3x_5 \oplus x_1x_3x_5 \oplus x_1x_3x_4 \oplus x_1x_2x_4 \oplus x_1x_2x_3 \\ =\; & x_5x_6 \cdot (x_4 \oplus x_2 \oplus x_1) \oplus a_1 \\ =\; & x_4x_6 \cdot (x_5 \oplus x_3 \oplus x_2) \oplus a_2, \end{aligned}$$

where each term of a_1 is not a multiple of x_5x_6 and each term of a_2 is not a multiple of x_4x_6. It follows from Theorem 15 that a linear right $*$-factor of h is a factor of

$$\gcd(x^4 \oplus x^2 \oplus x, x^5 \oplus x^3 \oplus x^2) = x \cdot (x^3 \oplus x \oplus 1).$$

It can be verified that actually $x_3 \oplus x_1 \oplus x_0$ is the largest linear right $*$-factor of h.

5.2 A new result on the candidate linear function m

In this subsection, we give another method to derive the candidate linear function m from the targeted function h whose theoretical analysis is different from that of [6].

For a Boolean function f, let us denote the unique function g such that

$$NL(f) = g \cdot x_n + r, n = \mathrm{ord}(NL(f)), \mathrm{ord}(r) < n$$

by $\Delta(f)$. Also $\Delta(f)$ can be seen as the quotient of $NL(f)$ divisible by x_n. Furthermore, for an positive integer e, define $\Delta^e(f) = \Delta\left(\Delta^{e-1}(f)\right)$ where $\Delta^0(f) = f$, composition of the function Δ. In particular, if e is the first nonnegative integer such that $\deg(\Delta^e(f)) < 2$, then define $\Delta^*(f) = \Delta^e(f)$.

Lemma 17 *Let $f, g \in \mathcal{C}^*$. Then* $\deg(f * g) \geq \deg(g)$. *Moreover, the equality holds if and only if* $\deg(f) = 1$.

Proof. If $\deg(f) = 1$, then it follows from Lemma 6 $\deg(f * g) = \deg(g)$. Suppose $\deg(f) > 1$. Then

$$NL(f) = \bigoplus_{k=2}^{\deg(f)} f_{[k]} \neq 0,$$

and so by Corollary 4 and Lemma 2 (ii), we have that $NL(f) * g \neq 0$ and $\deg(NL(f) * g) > \deg(g)$. Since

$$f * g = f_{[1]} * g \oplus NL(f) * g$$

and $\deg(f_{[1]} * g) = \deg(g)$ if $f_{[1]} \neq 0$, it can be seen that

$$\deg(f * g) = \deg(N_f * g) > \deg(g).$$

This completes the proof. ■

Theorem 18 *Let $h, f, l \in C^*$ such that $h = f * l$ and l is a linear Boolean function. Then*

$$\Delta(h) = \Delta(f) * l.$$

Proof. If $\deg(h) = 1$, then the result trivially holds. Assume $\deg(h) > 1$ in the following.

Let $\mathrm{ord}(NL(f)) = n$ and $\mathrm{ord}(l) = m$. Since $\deg(h) > \deg(l)$, it follows from Lemma 17 that $\deg(f) > 1$. Then f can be written

$$f = f_{[1]} \oplus NL(f) = f_{[1]} \oplus \Delta(f) \cdot x_n \oplus r,$$

where $\mathrm{ord}(r) < n$. Thus

$$h = f * l = f_{[1]} * l \oplus (\Delta(f) * l) \cdot (x_n * l) \oplus r * l. \tag{9}$$

Since $f_{[1]} * l$ is linear, $\mathrm{ord}(r * l) < \mathrm{ord}(x_n * l)$, and $\mathrm{ord}(\Delta(f) * l) < \mathrm{ord}(x_n * l)$, it follows from (9) that

$$\mathrm{ord}(NL(h)) = \mathrm{ord}(x_n * l) = n + m.$$

Hence

$$\Delta(h) = \Delta(f) * l.$$

This completes the proof. ■

The function $\Delta(h)$ in Theorem 18 may still be a nonlinear function. Then, recursively using Theorem 18 leads to the following result.

Corollary 19 *Let $h, f, l \in C^*$ such that $h = f * l$ and l is a linear Boolean function. Then*

$$\Delta^*(h) = \Delta^*(f) * l.$$

Obviously Corollary 19 tells us that $\Delta^*(h)$ is a multiple of the largest linear right $*$-factor of h.

Example 20 *Let h be as described in Example 16. Then it follows from Corollary 19 that*

$$\Delta^*(h) = x_4 \oplus x_2 \oplus x_1 = x \cdot (x^3 \oplus x \oplus 1)$$

is a multiple of the largest linear right $*$-factor of h.

The results of Theorems 15 and 18 can be used together, that is, the gcd of their results is also a desired candidate linear function. Although Theorems 15 and 18 often give precise answer for the largest linear right $*$-factor of a given function h, they are not sufficient. It still remains a problem how to directly identify the largest linear right $*$-factor of function h not just a multiple of that.

6 General Decomposition

Given a Boolean function $h \in \mathcal{C}^*$, in this section we shall show how to find all Boolean function pairs $(f, g) \in \mathcal{C}^* \times \mathcal{C}^*$ such that $h = f * g$. By Lemma 17, we know that $\deg(g)$ is bounded by $\deg(h)$. According to this observation, the first main idea of our decomposition algorithm is to search the desired function pairs (f, g) by the degree of g. A sketch of this main idea is illustrated in Table 1.

The second main idea of our decomposition algorithm is to reduce the general decomposition case (i.e., $d \geq 2$ in Table 1) to the simple case of decomposing a Boolean function h' into $h' = l * f$ where l is a linear Boolean function, which is completely solved in Section 4. Such reduction process is mainly illustrated in Subsection 6.1. A sketch of this main idea for solving S_d is illustrated in Table 2.

A small example is given in Appendix 7 to explain the decomposition ideas in this section.

In the following of this section, we assume that h is a characteristic function in $\mathcal{C}^*$ of degree greater than 1.

6.1 Find a candidate set for right $*$-factors

In this subsection, given an integer $d > 1$, we discuss how to find Boolean functions g such that $\deg(g) = d$ and g is a right $*$-factor of h. Note that the case $d = 1$ has been discussed

Table 1: Algorithm MAIN

Specification: $S \longleftarrow \text{MAIN}(h)$

Given: a Boolean function $h \in \mathcal{C}^*$

Find: a finite set S of Boolean function pairs $(f, g) \in \mathcal{C}^* \times \mathcal{C}^*$ such that $h = f * g$

begin

$S \longleftarrow \{(f, l) : h = f * l \text{ and } \deg(l) = 1\}$ (Please refer to Section 5)

for d **from** 2 **to** $\deg(h)$ **do**

find the set $S_d = \{(f, g) : h = f * g \text{ and } \deg(g) = d\}$

$S \longleftarrow S \cup S_d$

end for

return(S)

in Section 5.

Let $d > 1$ be a positive integer. We first generalize the symbol $\Delta(\cdot)$ defined in Subsection 5.2 to $\Delta_d(\cdot)$. For a Boolean function f, let us denote the unique function g such that

$$NL(f) = g \cdot x_n + r,$$

where $n = \text{ord}(f_{[\geq d]})$ and the polynomial r does not contain the variable x_n. Also $\Delta_d(f)$ can be seen as the quotient of $NL(f)$ divisible by x_n. Furthermore, for a positive integer e, define $\Delta_d^e(f) = \Delta_d\left(\Delta_d^{e-1}(f)\right)$ where $\Delta^0(f) = f$, composition of the function Δ_d. In particular, if e is the least nonnegative integer such that $\deg(\Delta_d^e(f)) < d$, then define $\Delta_d^*(f) = \Delta_d^e(f)$.

Example 21 *Let* $f = x_9 \oplus x_1x_9 \oplus x_6x_7x_8 \oplus x_4x_5x_6x_8 \oplus x_2x_4x_7 \oplus x_1$. *Then*

$$\Delta_3(f) = x_6x_7 \oplus x_4x_5x_6 \quad \textit{and} \quad \Delta_3^2(f) = x_7 \oplus x_4x_5.$$

Moreover, we have that $\Delta_3^*(f) = \Delta_3^2(f)$.

Table 2: Solve S_d for a given integer d

Specification: $S_d \longleftarrow$ Subalgorithm(h)

Given: a Boolean function $h \in \mathcal{C}^*$ and an integer $d > 2$

Find: a finite set S_d of Boolean function pairs $(f, g) \in \mathcal{C}^* \times \mathcal{C}^*$ such that

$h = f * g$ and $\deg(g) = d$

begin

Find a candiate set G such that if $h = f * g$, then $g \in G$. (Please refer to Subsections 6.1 and 6.2)

for $\hat{g} \in G$ **do**

find f such that $h = f * \hat{g}$ (f may not exist if $\hat{g}$ is not correct)(Please refer to Subsection 6.3)

if such f exists, then $S_d \longleftarrow S_d \cup (f, \hat{g})$

end for

return(S_d)

Theorem 22 *Let f, g, h be three Boolean functions such that $h = f * g$ and* $\deg(h) > \deg(g) = d > 2$. *If $g(x_0, \ldots, x_m) = g_0(x_0, \ldots, x_{m-1}) \oplus x_m$, then*

$$\Delta_{d+1}(h) = \Delta_2(f) * g \oplus r, \tag{10}$$

for some Boolean function r with $\deg(r) < d$.

Proof. Since $\deg(h) > \deg(g)$, it follows from Lemma 17 that $\deg(f) > 1$. Then f can be written

$$f = f_{[1]} \oplus NL(f) = f_{[1]} \oplus \Delta_2(f) \cdot x_n \oplus q,$$

where $\mathrm{ord}(NL(f)) = n$ and $\mathrm{ord}(q) < n$. Thus

$$\begin{aligned} h &= f * g \\ &= f_{[1]} * g \oplus (\Delta_2(f) \cdot x_n) * g \oplus q * g \\ &= f_{[1]} * g \oplus (\Delta_2(f) * g) \cdot x_{n+m} \oplus (\Delta_2(f) * g) \cdot (x_n * g_0) \oplus q * g. \end{aligned} \tag{11}$$

Let us denote

$$\begin{aligned} \lambda^1 &= (\Delta_2(f) * g) \cdot x_{n+m} \\ \lambda^2 &= (\Delta_2(f) * g) \cdot (x_n * g_0) \end{aligned}$$

Note that $\deg(f_{[1]} * g) = d$ by Lemma 17, which implies that $f_{[1]} * g$ is irrelavent to $\mathrm{ord}(h_{\geq[d+1]})$, and so

$$\mathrm{ord}(h_{\geq[d+1]}) = \mathrm{ord}(\lambda^1_{\geq[d+1]} + \lambda^2_{\geq[d+1]} + (q * g)_{\geq[d+1]}). \tag{12}$$

On one hand, it can be seen that

$$\mathrm{ord}(\lambda^2) < n + m \text{ and } \mathrm{ord}(q * g) < n + m. \tag{13}$$

On the other hand, since by Lemma 17 $\deg(\Delta_2(f) * g) > d$ and it is clear that $\mathrm{ord}(\Delta_2(f) * g) < n + m$, it follows that

$$\mathrm{ord}(\lambda^1_{\geq[d+1]}) = \mathrm{ord}(\lambda^1) = n + m. \tag{14}$$

Hence, it follows from (12), (13) and (14) that

$$\mathrm{ord}(h_{\geq[d+1]}) = n + m.$$

Consequently, we deduce from (11) that

$$\Delta_{d+1}(h) = \Delta_2(f) * g \oplus r,$$

where the Boolean function r satisfies that

$$f_{[1]} * g = r \cdot x_{n+m} + r',$$

the polynomial r' does not contain the variable x_{n+m}. ■

Remark 23 *Let h, f, g be as described in Theorem 22. It is easy to see that Theorem 22 also holds for $h = f * g \oplus p$ if $\deg(p) < d$, since p is irrelevant to $h_{\geq[d+1]}$.*

Suppose $h = f * g$ for some $h, f, g \in \mathcal{C}^*$. According to Remark 23, one could recursively using Theorem 22 until the degree of the Boolean function on the left side of the equality (10) is less than $d+1$, which results in

$$\Delta^*_{d+1}(h) = \Delta^*_2(f) * g \oplus r \tag{15}$$

for some Boolean function r with $\deg(r) < d$. Since $\deg(r) < d$ and $\Delta^*_2(f)$ is a linear Boolean function, the equality (15) implies the following result.

Corollary 24 *Let f, g, h be as described in Theorem 22. Then*

$$\left(\Delta^*_{d+1}(h)\right)_{[d]} = \Delta^*_2(f) * g_{[d]}.$$

Actually, Corollary 24 tells us that we could retrieve $g_{[d]}$ from $\left(\Delta^*_{d+1}(h)\right)_{[d]}$ which is the case discussed in Section 4. Once we get $g_{[d]}$, let us write (15) as

$$\Delta^*_{d+1}(h) = \Delta^*_2(f) * g_{[d]} \oplus \Delta^*_2(f) * (g_{[d-1]} \oplus \cdots \oplus g_{[1]}) \oplus r,$$

and so

$$\Delta^*_{d+1}(h) \oplus \Delta^*_2(f) * g_{[d]} = \Delta^*_2(f) * (g_{[d-1]} \oplus \cdots \oplus g_{[1]}) \oplus r.$$

Note that the Boolean function on the left side of the above equality is known. If $r_{[d-1]}$ is also known, then it is clear that

$$\Delta^*_{d+1}(h) \oplus \Delta^*_2(f) * g_{[d]} \oplus r_{[d-1]} = \Delta^*_2(f) * g_{[d-1]},$$

and so we could retrieve $g_{[d-1]}$ by decomposing $\Delta^*_{d+1}(h) \oplus \Delta^*_2(f) * g_{[d]} \oplus r_{[d-1]}$. Similarly, if we know $r_{[d-2]}$, then we could retrieve $g_{[d-2]}$, and recursively if we know $r_{[i]}$, then we could retrieve $g_{[i]}$ for $i = d-3, \ldots, 1$. Now the key problem is how to reasonably guess $r_{[d-1]}, r_{[d-2]}, \ldots, r_{[1]}$. We discuss this problem in the next subsection.

6.2 How to reasonably guess $r_{[d-1]}, r_{[d-2]}, \ldots, r_{[1]}$

In this subsection we shall show that $r_{[i]}$ is determined by $g_{[i+1]}, \ldots, g_{[d]}$ for $i = d-1, \ldots, 1$.

Before we come back to the proof of Theorem 22, let us introduce a notation. Let f be a Boolean function and t a term. Then we use $\langle \frac{f}{t} \rangle$ to denote the quotient of f divisible by t, i.e.,

$$f = \langle \frac{f}{t} \rangle \cdot t + q,$$

where each term of q is not a multiple of t.

Let f, g be as described in Theorem 22. Then according to the proof of Theorem 22, the Boolean function r in (10) is actually given by

$$r = \langle \frac{f_{[1]} * g}{x_{n+m}} \rangle$$

and so we rewrite (10) in the following way

$$\Delta_{d+1}(h) = \Delta_2(f) * g \oplus \langle \frac{f_{[1]} * g}{x_{n+m}} \rangle.$$

Let us denote $f_{[1]} = l_0$ and the linear part of $\Delta_2^i(f)$ by l_i for any positive integer i. Assume $\Delta_{d+1}^*(h) = \Delta_{d+1}^e(h)$. Then the process of recursively using Theorem 22 could be illustrated by the following equalities:

$$\begin{aligned}
\Delta_{d+1}(h) &= \Delta_2(f) * g \oplus \langle \frac{l_0 * g}{x_{i_0}} \rangle \\
\Delta_{d+1}^2(h) &= \Delta_2^2(f) * g \oplus \langle \frac{l_1 * g}{x_{i_1}} \rangle \oplus \langle \frac{l_0 * g}{x_{i_0} x_{i_1}} \rangle \\
&\cdots \\
\Delta_{d+1}^e(h) &= \Delta_2^e(f) * g \oplus \langle \frac{l_{e-1} * g}{x_{i_{e-1}}} \rangle \oplus \cdots \oplus \langle \frac{l_1 * g}{x_{i_0} \cdots x_{i_{e-1}}} \rangle \oplus \langle \frac{l_0 * g}{x_{i_0} \cdots x_{i_{e-1}}} \rangle \qquad (16)
\end{aligned}$$

where $i_j = \mathrm{ord}((\Delta_{d+1}^j(h))_{\geq [d+1]})$ for $j = 0, 1, \ldots, e-1$. Since $\deg(l_i * g) = \deg(g) = d$, if follows from (17) that

$$r_{[d-1]} = \langle \frac{l_{e-1} * g_{[d]}}{x_{i_{e-1}}} \rangle$$

and so $r_{[d-1]}$ is only relevant to $g_{[d]}$. Consequently, a reasonable range for $\mathrm{ord}(l_{e-1})$ is given by

$$i_{e-1} - \mathrm{ord}(g_{[d]}) \leq \mathrm{ord}(l_{e-1}) \leq \mathrm{ord}(\Delta_{d+1}^{e-1}(h)) - \mathrm{ord}(g_{[d]})^1.$$

and so

$$T(r_{[d-1]}) \in \bigcup_{i=i_{e-1}-\mathrm{ord}(g_{[d]})}^{\mathrm{ord}(\Delta_{d+1}^{(e-1)}(h))-\mathrm{ord}(g_{[d]})} T(\langle \frac{x_i * g_{[d]}}{x_{i_{e-1}}} \rangle).$$

Hence we could guess $r_{[d-1]}$ within this range. The rest $r_{[d-2]}, \ldots, r_{[1]}$ could be guessed similarly, and we omit such tedious discussion for them.

6.3 How to find f such that $h = f * g$ if g is known

Recall that Subsections 6.1 and 6.2 discuss how to find a candidate set of right $*$-factors g of h. To complete the search for Boolean function pairs (f, g) such that $h = f * g$, in this subsection, we discuss how to find the corresponding left $*$-factor f if a right $*$-factor g of h is given. Assume $\mathrm{ord}(g) = m$.

The key observation is simple and follows from Lemma 2. Suppose there is a function f such that $h = f * g$ and $f = t_1 \oplus t_2 \oplus \cdots \oplus t_k$, where $t_1 \prec t_2 \prec \cdots \prec t_k$ are terms. Then

$$h = f * g = t_1 * g \oplus t_2 * g \oplus \cdots \oplus t_k * g. \tag{17}$$

Since by Lemma 2 we have

$$\mathrm{HT}(t_i * g) = x_m * t_i \text{ for } 1 \leq i \leq k,$$

it follows from (17) that

$$\mathrm{HT}(h) = x_m * \max\{t_1, t_2, \ldots, t_k\} = x_m * t_k.$$

This shows that we can easily compute $\mathrm{HT}(f) = t_k$ from $\mathrm{HT}(h)$. Then let

$$h' = h \oplus x_m * g.$$

[1] If $i_{e-1} - \mathrm{ord}(g_{[d]}) > \mathrm{ord}(l_{e-1})$, then $\mathrm{ord}(g_{[d]}) + \mathrm{ord}(l_{e-1}) < i_{e-1}$, and so $r_{[d-1]} = 0$

Consequently, we have

$$h' = t_1 * g \oplus t_2 * g \oplus \cdots \oplus t_{k-1} * g,$$

and so t_{k-1} can be restored from $\mathrm{HT}(h')$ similarly. Analogously, we could easily restore $t_{k-2}, \ldots, t_1$, and so is f.

Finally we remark that if a given function g is a wrong guess, then the above process to compute f would break off at some point. Usually the scale of the intermediate functions appeared during the above process, say h', $h'' = h' \oplus t_{k-1} * g$, etc., will increase dramatically if g is a wrong guess, while that will decrease gradually if g is a right guess. Thus, setting a bound, say $2 \cdot |T(h)|$, for the number of terms of intermediate functions can save much time and will not affect the result for the most of the cases.

6.4 Complexity analysis and experimental result

Like many other algorithms related to NFSRs, the precise complexity formula of the general decomposition algorithm proposed in this section is hard to give. But it is easy to see that the complexity of the algorithm is related to the ANF of the function being decomposed, i.e., the number of terms, the degree, and the order. Generally, as the number of terms or the degree or the order increases, the algorithm needs more time to terminate. This is also shown by the following experimental results listed in Tables 3, 4, 5, and 6.

We implemented the whole algorithm by the symbolic computation software Singular except the subalgorithm used for decomposition into a cascade connection of an NFSR into an LFSR, because this subalgorithm has been well studied in [6]. All the experiments were done on a PC with Intel Core i5 CPU 2.4 GHZ and 8.0 GB RAM, and the experimental results are listed in Tables 3, 4, 5, and 6. The data of the first three tables correspond to Experiment 25 whose aim is to show the impact of the order and the number of terms of the function being decomposed on running time. It can be seen that the running time increases as either the order or the number of terms increases. The data of the last table

correspond to Experiment 26 whose aim is to show the impact of parameter d described in Table 1 on running time by comparing with the data of Experiment 25, where $d = 2$ in Experiment 25 while $d = 3$ in Experiment 26. It can be seen that the running time increases as d increases. Overall, it is found that the parameter d has the greatest impact on the running time. This is because given d, our algorithm is an iterative algorithm and the number of iterations is $d-1$, which increases as d increases. Moreover, the more number of iterations probably leads to more guesses that need to be verified. Since d is bounded by $\deg(h)$, it follows that $\deg(h)$ has the greatest impact on running time of the proposed algorithm compared to the order of h and the number of terms of h.

Experiment 25 *We remark that for a random characteristic function h, the probability that h could be decomposed into $h = f * g$ for some characteristic functions f and g is very small. Thus, instead of randomly generating h, we randomly generate f and g and decompose $h = f * g$ use our algorithm. Given parameters n, a, b which are positive integers, the experimental steps are as follows.*

Step 1 *We randomly generate a sufficient number of pairs of quadratic characteristic functions (f_i, g_i) of order n each and compute $h_i = f_i * g_i$.*

Step 2 *Among the functions generated in Step 1, only keep 1000 instances of h_i satisfying $a \leq |T(h_i)| < b$ and discard all the others. Here we control the number of terms of h_i within a certain prescribed range to show the relation between the number of terms of the function being decomposed and the running time of our algorithm.*

Step 3 *For $1 \leq i \leq 1000$, decompose h_i retained in Step 2 using our Singular program to find f_i and g_i by directly setting the parameter d to be 2 (see Table 1 for the implication of the notation d).*

Experiment 26 *For each $n \in \{20, 30, 40\}$, the experimental steps are as follows.*

Table 3: Results for Experiment 25 with $a = 0$ and $b = 100$

ord(f) & ord(g)	ord(h)	Number of samples	Average Number of $\lvert T(h)\rvert$	Average Time
20	40	1000	56.6	14.8 seconds
30	60	1000	56.2	24.3 seconds
40	80	1000	56.5	40.0 seconds

Table 4: Results for Experiment 25 with $a = 100$ and $b = 200$

ord(f) & ord(g)	ord(h)	Number of samples	Average Number of $\lvert T(h)\rvert$	Average Time
20	40	1000	140	38.4 seconds
30	60	1000	140.1	62.1 seconds
40	80	1000	141.8	98.3 seconds

Step 1 *We randomly generate a sufficient number of pairs of functions* (f_i, g_i) *with* $\deg(f_i) = 2$ *and* $\deg(g_i) = 3$ *of order* n *each and compute* $h_i = f_i * g_i$.

Step 2 *Among the functions generated in Step 1, only keep 100 instances of* h_i *satisfying* $|T(h_i)| < 300$ *and discard all the others.*

Step 3 *For* $1 \leq i \leq 100$, *decompose* h_i *retained in Step 2 using our Singular program to find* f_i *and* g_i *by directly setting the parameter* d *to be 3 (see Table 1 for the implication of the notation* d*).*

Remark 27 *The average running time for* $\text{ord}(h) = 80$ *in Table 4 is a bit greater than that in Table 5. This is because a few instances of the 1000 samples corresponding to* $100 \leq |T(h_i)| < 200$ *happened to be difficult to decompose. This implies that not only the*

Table 5: Results for Experiment 25 with $a = 200$ and $b = 300$

ord(f) & ord(g)	ord(h)	Number of samples	Average Number of $\lvert T(h)\rvert$	Average Time
20	40	1000	239	64.7 seconds
30	60	1000	239.8	74.1 seconds
40	80	1000	242.4	92.5 seconds

Table 6: Results for Experiment 26

ord(f) & ord(g)	ord(h)	Number of samples	Average Number of $\lvert T(h)\rvert$	Average Time
20	40	100	87	3.2 minutes
30	60	100	98	4.8 minutes
40	80	100	88	9.5 minutes

number of terms but also the distribution of terms has some impact on running time, whose influence is hard to predict.

Remark 28 *Most of functions decomposed in Experiment 25 have degree 4, and most of h decomposed in Experiment 26 have degree 6.*

7 Conclusion

In this paper we study the problem of decomposing an NFSR into a cascade connection of two smaller NFSRs. Only a very special case of this problem has been solved before, i.e., the cascade connection of an NFSR into an *LFSR*. We propose an algorithm to completely solve this decomposition problem. Experimental results show that the proposed algorithm

is practical.

The uniqueness of decomposing an NFSR into a cascade connection is till open. It seems that such decomposition is unique for most of the functions but no proof has been found. This is an interesting theoretical problem and will be one subject of our future work.

Appendix: An Auxiliary Theorem

Theorem 29 *Let $h \in \mathcal{C}^*$. If $h = f * g$ for some functions $f, g \in \mathcal{C}$, then there are two functions $f, g \in \mathcal{C}^*$ such that $h = f * g$.*

Proof. Assume $f \in \mathcal{B}_n$ and $g \in \mathcal{B}_m$, where n, m are positive integers. First it can be easily seen that the case $f(0, \ldots, 0) = 1$ and $g(0, \ldots, 0) = 0$ is impossible. Thus, we have two cases to be discussed.

If $f(0, \ldots, 0) = 0$ and $g(0, \ldots, 0) = 1$, then we write $g = g' \oplus 1$. Consequently, we get

$$\begin{aligned} h &= f * (g' \oplus 1) \\ &= f(x_0 \oplus 1, \ldots, x_{n-1} \oplus 1) * g'. \end{aligned} \tag{18}$$

Let us denote $f' = f(x_0 \oplus 1, \ldots, x_{n-1} \oplus 1)$. Since $h(0, \ldots, 0) = 0$, it follows from (18) that $f'(0, \ldots, 0) = 0$. Thus, we have $h = f' * g'$ and $f', g' \in \mathcal{C}^*$.

If $f(0, \ldots, 0) = 1$ and $g(0, \ldots, 0) = 1$, then we write $f = f' \oplus 1$ and $g = g' \oplus 1$. With these notations, we get

$$\begin{aligned} h &= (f' \oplus 1) * (g' \oplus 1) \\ &= (f' * (g' \oplus 1)) \oplus 1 \\ &= (f'(x_0 \oplus 1, \ldots, x_{n-1} \oplus 1) * g') \oplus 1 \end{aligned} \tag{19}$$

Let us denote $f' = f(x_0 \oplus 1, \ldots, x_{n-1} \oplus 1)$. Since $h(0, \ldots, 0) = 0$ and $g'(0, \ldots, 0) = 0$, it follows from (19) that $f'(0, \ldots, 0) = 1$, and so let us further denote $f' = f'' \oplus 1$. Thus, (19)

implies that

$$h = ((f'' \oplus 1) * g') \oplus 1 = f'' \oplus g'.$$

■

Appendix: A Small Example

In this section, we give a small example to illustrate the discussions of Subsections 6.1, 6.2, and 6.3.

Let

$$\begin{aligned} h \;=\; & x_8 \oplus x_4x_5x_7 \oplus x_5x_7 \oplus x_3x_4x_7 \oplus x_4x_7 \oplus x_3x_7 \oplus x_2x_7 \oplus x_1x_7 \oplus x_7 \oplus x_3x_4x_5x_6 \\ & \oplus x_3x_5x_6 \oplus x_2x_5x_6 \oplus x_1x_5x_6 \oplus x_3x_6 \oplus x_6 \oplus x_2x_5 \oplus x_1x_5 \oplus x_1x_3 \oplus x_2 \oplus x_0. \end{aligned}$$

We shall find (f, g) such that $h = f * g$ and $\deg(g) = 2$.

First, we compute

$$\Delta_3(h) = x_4x_5 \oplus x_5 \oplus x_3x_4 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1,$$

and it is clear that $\Delta_3^*(h) = \Delta_3(h)$. Then we know that

$$x_4x_5 \oplus x_3x_4 = \Delta_2(f) * g_{[2]}.$$

Thus

$$g_{[2]} \in \{x_1x_2, x_2x_3\}.$$

Case 1 If $g_{[2]} = x_1x_2$, then $\Delta_2(f) = x_3 \oplus x_2$. It follows that

$$r_{[1]} = 0 \;\text{ or }\; r_{[1]} = \langle \frac{x_5 * g_{[2]}}{x_7} \rangle = x_6.$$

If $r_{[1]} = 0$, then

$$x_5 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 = \Delta_2(f) * g_{[1]},$$

which does not hold since the least subscript of the variables in $\Delta_2(f) * g_{[1]}$ is at least 2. If $r_{[1]} = x_6$, then

$$x_6 \oplus x_5 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 = \Delta_2(f) * g_{[1]},$$

which does not hold since $x^6 \oplus x^5 \oplus x^4 \oplus x^3 \oplus x^2 \oplus x$ is not divisible by $\phi(\Delta_2(f))$ in $\mathbb{F}_2[x]$. Hence $g_{[2]} \neq x_1x_2$.

Case 2 If $g_{[2]} = x_2x_3$, then $\Delta_2(f) = x_2 \oplus x_1$. It follows that

$$r_{[1]} = 0 \quad \text{or} \quad r_{[1]} = \langle \frac{x_4 * g_{[2]}}{x_7} \rangle = x_6.$$

If $r_{[1]} = 0$, then there is a contradiction the same as in Case 1. If $r_{[1]} = x_6$, then

$$x_6 \oplus x_5 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 = \Delta_2(f) * g_{[1]},$$

and so

$$g_{[1]} = x_4 \oplus x_2 \oplus x_0.$$

Thus we get

$$g = x_4 \oplus x_2x_3 \oplus x_2 \oplus x_0.$$

Then we solve f such that $h = f * g$. Since $\mathrm{HT}(h) = x_8$, we know $\mathrm{HT}(f) = x_4$. Set

$$\begin{aligned} h^{(1)} &= h \oplus x_4 * g \\ &= x_6x_7 \oplus x_4x_5x_7 \oplus x_5x_7 \oplus x_3x_4x_7 \oplus x_4x_7 \oplus x_3x_7 \oplus x_2x_7 \oplus x_1x_7 \oplus x_7 \oplus x_3x_4x_5x_6 \\ &\quad \oplus x_3x_5x_6 \oplus x_2x_5x_6 \oplus x_1x_5x_6 \oplus x_3x_6 \oplus x_2x_5 \oplus x_1x_5 \oplus x_4 \oplus x_1x_3 \oplus x_2 \oplus x_0. \end{aligned}$$

Since $\mathrm{HT}(h^{(1)}) = x_6x_7$, we know $x_2x_3 \in T(f)$. Set

$$\begin{aligned} h^{(2)} &= h^{(1)} \oplus (x_2x_3) * g \\ &= x_5x_7 \oplus x_3x_4x_7 \oplus x_3x_7 \oplus x_1x_7 \oplus x_7 \oplus x_3x_4x_5x_6 \oplus x_3x_5x_6 \\ &\quad \oplus x_1x_5x_6 \oplus x_3x_4x_5 \oplus x_1x_5 \oplus x_3x_4 \oplus x_4 \oplus x_2x_3 \oplus x_1x_3 \oplus x_2 \oplus x_0. \end{aligned}$$

Since $\mathrm{HT}(h^{(2)}) = x_5x_7$, we know $x_1x_3 \in T(f)$. Set

$$\begin{aligned} h^{(3)} &= h^{(2)} \oplus (x_1x_3) * g \\ &= x_7 \oplus x_5x_6 \oplus x_5 \oplus x_4 \oplus x_2x_3 \oplus x_3 \oplus x_2 \oplus x_0. \end{aligned}$$

Finally, it is easy to see that

$$h^{(3)} = (x_3 \oplus x_0) * g,$$

and so we get

$$f = x_4 \oplus x_2x_3 \oplus x_1x_3 \oplus x_3 \oplus x_0.$$

It is can be verified that

$$h = (x_4 \oplus x_2x_3 \oplus x_1x_3 \oplus x_3 \oplus x_0) * (x_4 \oplus x_2x_3 \oplus x_2 \oplus x_0),$$

and so (f, g) is the right answer.

References

[1] M. Hell, T. Johansson, and W. Meier, "The Grain family of stream ciphers," in *New Stream Cipher Designs: The eSTREAM Finalists*, LNCS 4986, pp. 179–190. New York: Springer-Verlag, 2008.

[2] C. Cannière and B. Preneel, "Trivium," in *New Stream Cipher Designs: The eSTREAM Finalists*, LNCS 4986, pp. 244–266. New York: Springer-Verlag, 2008.

[3] C. Cannière, O. Dunkelman, and M. Knežević, "KATAN and KTANTAN–A family of small and efficient hardware-oriented block ciphers," in: C. Clavier and K. Gaj (Eds.) *CHES 2009*, LNCS 5747, pp. 272–288. Berlin: Springer-Verlag, 2009,

[4] D.H. Green and K.R. Dimond, "Nonlinear product-feedback shift registers," *PROC. IEE*, vol.117, no. 4, pp. 681–686, 1970.

[5] Z. Ma, W.F. Qi, and T. Tian, "On the decomposition of an NFSR into the cascade connection of an NFSR into an LFSR," *J. Complex.*, vol. 29, no. 2, pp. 173–181, 2013.

[6] J.M. Zhang, W.F. Qi, T. Tian, and Z.X. Wang, "Further results on the decomposition of an NFSR into the cascade connection of an NFSR into an LFSR," *IEEE Trans. Inf. Theory*, vol.61, no.1, pp. 645-654, 2015.

[7] S.W. Golomb, *Shift Register Sequences*, Aegean Park Press, California, 1982.

[8] J. Mykkeltveit, M.K. Siu, and P. Tong, "On the cycle structure of some nonlinear shift register sequences," *Information and Control*, vol.43, pp. 202–215, 1979.