On Decomposition of an NFSR into a Cascade Connection of Two Smaller NFSRs*

Tian Tian and Wen-Feng Qi[†][‡]

July 9, 2014

Abstract

Nonlinear feedback shift registers (NFSRs) are an important type of sequence generators used for building stream ciphers. The shift register used in Grain, one of eSTREAM finalists, is a cascade connection of two NFSRs, which is also known as nonlinear product-feedback shift registers proposed in 1970. This paper provides a series of algorithms to decompose a given NFSR into a cascade connection of two smaller NFSRs. By decomposing an NFSR into a cascade connection of two smaller NFSRs, some properties regarding cycle structure of the original NFSR could be known.

Keywords: stream ciphers, nonlinear feedback shift registers, cascade connection, Grain

Mathematics Subject Classifications (2000): 94A55, 94A60

*This work was supported by the National Natural Science Foundation of China (Grant 61100202, 61272042).

[†] Tian Tian and Wen-Feng Qi are with the department of Applied Mathematics, Zhengzhou Information Science and Technology Institute, Zhengzhou, P.R.China. (e-mail: tiantian_d@126.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. Nonlinear feedback shift registers (NFSRs) are an important type of nonlinear sequence generators with more than 40 years of research. Grain and Trivium, two eSTREAM hardware-oriented finalists, use NFSRs as a main building block, see [1, 2]. Despite that NFSRs are frequently appeared in stream cipher designs and have a quite long time of research, many algebraic properties of NFSRs are essentially unknown. In this paper we are concerned with a type of decomposition of NFSRs proposed in 1970 [3], namely cascade connections of two NFSRs.

Let $f_1(x)$ and $f_2(x)$ be two polynomials over $\mathbf{F}_2$, the finite field of two elements. It is known that a sequential circuit made up from a cascade connection of the LFSR with characteristic polynomial $f_1(x)$ into the LFSR with characteristic polynomial $f_2(x)$ outputs the same family of sequences as the LFSR with characteristic polynomial $f_1(x)f_2(x)$ [3]. Thus a product LFSR (an LFSR with a composite characteristic polynomial) can be interpreted as a cascade connection of its factors.

In [3] the author demonstrated such 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). For example, the nonlinear primitive used in Grain is a cascade connection of a 80-stage LFSR into a 80-stage NFSR. It was shown in [3] that a cascade connection of the NFSR F_1 with characteristic function $f_1(x_0, x_1, \ldots, x_n)$ into the NFSR F_2 with characteristic function $f_2(x_0, x_1, \ldots, x_m)$ outputs the same family of sequences as the NFSR F_3 with characteristic function $f_1 * f_2$. Moreover, if the NFSR F_1 can generate all zero sequence, i.e., $f_1(0, 0, \ldots, 0) = 0$, then the family of outputting sequences of F_2 is a subset of that of F_3. In such case, we say that F_1 is a factor of F_3 and F_3 is reducible, as what we do in the LFSR domain. As for cryptographic applications, an NFSR is expected to be irreducible or at least the outputting sequences used to produce keystreams should not drop into the subfamily of sequences generated by one of its factor NFSRs. So far how to do this is still open.

In [4], the authors studied a very special decomposition case of $*$-product, that is, the decomposition of an NFSR into the cascade connection of an NFSR into an LFSR. In this paper, we consider the general decomposition problem. We present a series of algorithms to decompose an NFSR F into a cascade connection of two others F_1 and F_2, where F, F_1, F_2 could output the all-zero sequence. For the special case that F_2 is an LFSR, our algorithms are similar with those given in [4] but not the same. Combining the two ideas could lead to a better algorithm for the decomposition of an NFSR into the cascade connection of an NFSR into an LFSR. The theories behind the algorithms are elementary and the computation involved in the algorithms are simple. Generally the algorithms are efficient for an NFSR whose characteristic function is sparse and has small degree, though the time complexity of the worst case is exponential.

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 $\mathbf{F}_2^n$ into $\mathbf{F}_2$ 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 $u_\alpha \in \mathbf{F}_2$, 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

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

Usually it is more convenient for us to write a term $x_0^{\alpha_0} x_1^{\alpha_1} \cdots x_{n-1}^{\alpha_{n-1}}$ in the form $x_{i_1} x_{i_2} \cdots x_{i_k}$, and we always assume that $i_1 < i_2 < \cdots < i_k$.

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 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. Next, we extend the term order to an order on $\mathcal{B}_n$. Let $f, g \in \mathcal{B}_n$. Then we define $f \preceq g$ if and only if $f = g$ or $\mathrm{HT}(f \oplus g) \in T(g)$. Moreover, we write $f \prec g$ if $f \preceq g$ and $f \neq g$.

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, denoted by $f \parallel_L h$ and $g \parallel_R h$ respectively. 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) $f * g = \bigoplus_{t \in T(f)} (t * g)$;

(iii) $(f \oplus 1) * g = f * g \oplus 1$;

(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.

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

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

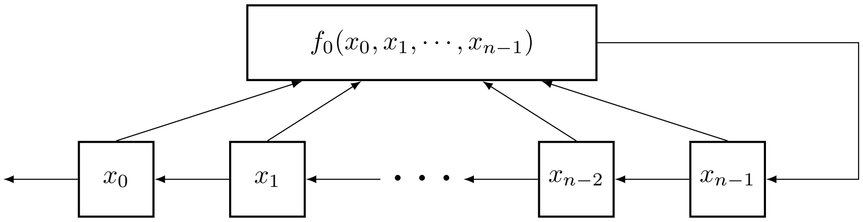

Figure 1: An n-stage NFSR

The function ϕ maps a linear Boolean function to a univariate polynomial over $\mathbf{F}_2$, which is a one-to-one correspondence. It can be seen that for two linear Boolean functions f, g, $\phi(f * g) = \phi(f)\phi(g)$.

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) 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 [5, 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

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

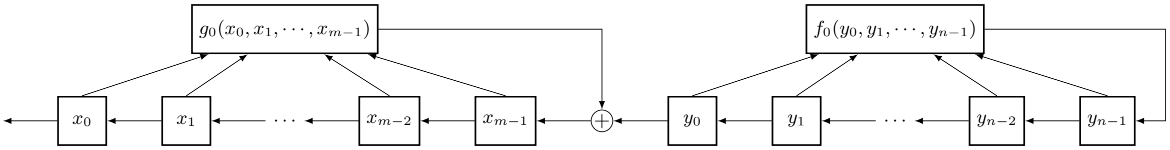

Figure 2: The cascade connection of NFSR (f) into NFSR(g)

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

Let $m \in \mathbb{N}^*$ and $g(x_0, x_1 \dots, x_m) = g_0(x_0, x_1, \dots, x_{m-1}) \oplus x_m \in \mathcal{B}_{m+1}$. 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 $y_0, y_1, \dots, y_{n-1}$. 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 [3] and [6]. Here the notation $f * g$ is the same as what D.H. Green and K.R. Dimond in [3] called the product of f and g denoted by $f \cdot g$.

In the following, we consider how to decompose an NFSR in $\mathcal{C}^*$ into a cascade connection of two others in $\mathcal{C}^*$. Thus, all factorizations w.r.t. $*$-product are assumed to be in $\mathcal{C}^*$ in the following paper.

3 Algorithms

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 $h = f * g$. A sketch of the main idea is illustrated in Table 1.

As a preparation, in Subsection 3.1 we derive some properties of Boolean functions w.r.t. $*$-product. In Subsection 3.2, 3.3, and 3.4, we discuss the specific subalgorithms to implement the algorithm MAIN of Table 1. In Table 2, we list all the subalgorithms and their functions appearing in the following.

```
**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
$S \longleftarrow \varnothing$
for $d$ from $1$ to $\deg(h)$ do
    find a set $S_d$ of Boolean function pairs $(f, g)$ such that $h = f * g$
    and $\deg(g) = d$
    $S \longleftarrow S \cup S_d$
end for
return($S$)
```

Table 1. Algorithm MAIN

Algorithm Name	Algorithm Function
FIND-RIGHT-FACTOR(h, f)	Given $h \in \mathcal{C}^*$ and a linear function f, find $g \in \mathcal{C}^*$ such that $h = f * g$.
EQUAL-DEGREE(h)	Given $h \in \mathcal{C}^*$, find all $(f, g) \in \mathcal{C}^* \times \mathcal{C}^*$ such that $h = f * g$ and $\deg(h) = \deg(g)$.
FIND-LEFT-LINEAR(h, g)	Given $h, g \in \mathcal{C}^*$ with $\deg(h) = \deg(g)$, find $f \in \mathcal{C}^*$ such that $h = f * g$.
FIND-LEFT-FACTOR(h, g)	Given $h, g \in \mathcal{C}^*$, find $f \in \mathcal{C}^*$ such that $h = f * g$.
GIVENDEG(h, d) (together with SUBALG-FOR-GIVENDEG)	Given $h \in \mathcal{C}^*$ and $d \in \mathbb{N}^*$ with $d \leq \deg(h)$, find all $(f, g) \in \mathcal{C}^* \times \mathcal{C}^*$ such that $h = f * g$ and $\deg(g) = d$.

Table 2. A list of Subalgorithms for MAIN

3.1 Theoretical Bases

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* $k \geq 1$. *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 corollary.

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

As for a linear left $*$-factor, the requirement on the form of the function g in Lemma 2 can be relaxed and the uniqueness can be extended to right $*$-factors.

Lemma 6 *Let h, g be two Boolean functions which are not constants, and let f be a linear Boolean function with $\mathrm{ord}(f) = n$. If $h = f * g$, then $\deg(h) = \deg(g)$, $h_{[i]} = f * g_{[i]}$, and $\mathrm{HT}(h_{[i]}) = x_n * \mathrm{HT}(g_{[i]})$ for $1 \leq i \leq \deg(h)$.*

Proof. Since f is linear, it can be seen that $f * t$ is a homogenous Boolean function of degree $\deg(t)$ for any term $t \neq 1$ and $\mathrm{HT}(f * t) = x_n * t$. Then this and the fact $f * g = \bigoplus_{t \in T(g)} f * t$ imply that the lemma holds. ■

Lemma 7 *Let $h \in \mathcal{C}^*$ and f a linear Boolean function. If $f \parallel_L h$, then there exists a unique Boolean function $g \in \mathcal{C}^*$ such that $h = f * g$.*

Proof. Suppose there is another Boolean function $g' \in \mathcal{C}^*$ and $g' \neq g$ such that $h = f * g'$. Then by Proposition (iv)

$$0 = f * g \oplus f * g' = f * (g \oplus g'),$$

a contradiction to Lemma 6. ■

Lemma 8 *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 by Lemma 6 $\deg(f * g) = \deg(g)$. Suppose $\deg(f) > 1$. Then

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

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

$$f * g = f_{[1]} * g \oplus N_f * g$$

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. ■

3.2 Find right $*$-factors of h of the same degree with h

Given $h \in \mathcal{C}^*$ with $\deg(h) = d$, in this subsection, we discuss how to find all Boolean function pairs $(f, g) \in \mathcal{C}^* \times \mathcal{C}^*$ such that $h = f * g$ and $\deg(h) = \deg(g)$. By Lemma 8, we know that f must be linear in such case.

First, suppose we know a linear Boolean function f with $\mathrm{ord}(f) = n$ such that $f \|_L h$. We discuss how to determine g satisfying that $h = f * g$. By Lemma 6, we have that

$$h_{[i]} = f * g_{[i]} \text{ for } 1 \leq i \leq d.$$

This shows that $g_{[i]}$ is only related to $h_{[i]}$ and f for $1 \leq i \leq d$, and so $g_{[1]}, g_{[2]}, \ldots, g_{[d]}$ can be solved independently. Assume $1 \leq i \leq d$ is fixed. Since by Lemma 6

$$\mathrm{HT}(h_{[i]}) = x_n * \mathrm{HT}(g_{[i]}), \tag{7}$$

it follows that if $h_{[i]} \neq 0$, then

$$\mathrm{HT}(g_{[i]}) = x_{j_1 - n} x_{j_2 - n} \cdots x_{j_i - n}$$

where $\mathrm{HT}(h_{[i]}) = x_{j_1} x_{j_2} \cdots x_{j_i}$; otherwise, $g_{[i]} = 0$. This means that $\mathrm{HT}(g_{[i]})$ can be easily determined from h and f. Then set

$$h_{[i]}^{(1)} = h_{[i]} \oplus \big(f * \mathrm{HT}(g_{[i]})\big) \quad \text{and} \quad g_{[i]}^{(1)} = g_{[i]} \oplus \mathrm{HT}(g_{[i]}).$$

It follows from (7) that

$$h_{[i]}^{(1)} = f * g_{[i]}^{(1)},$$

and so $\mathrm{HT}(g_{[i]}^{(1)})$ can be derived from $\mathrm{HT}(h_{[i]}^{(1)})$. Continuing this process, it can be seen that $g_{[i]}$ can be solved term by term. Based on this idea, we give the following Theorem 9. We remark that since the procedures of solving $g_{[1]}, g_{[2]}, \ldots, g_{[d]}$ are independent, we can solve $g_{[1]}, g_{[2]}, \ldots, g_{[d]}$ in any order.

For a positive integer k and k terms $t_1, t_2, \ldots, t_k$, let us denote by $\min\{t_1, t_2, \ldots, t_k\}$ the minimum term w.r.t. $\preceq$ among $t_1, t_2, \ldots, t_k$.

Theorem 9 *Let $h \in \mathcal{C}^*$ and f a linear Boolean function. Then the algorithm* FIND-RIGHT-FACTOR *of Table* 3 *computes the unique Boolean function $g \in \mathcal{C}^*$ such that $h = f * g$ if g exists.*

$h = \quad h \quad g$

$^{\mathrm{i}} = hg^{1(}$

[7]TF157.97Tf2.35E0[()]TF11116o583.290n.86Td[(g)]TF911.964-218d[()[()]TF11116o5g

Proof. Suppose there are $N \in \mathbb{N}^* \cup \{\infty\}$ runs through the while-loop, where $h^{(i)}$ is converted into $h^{(i+1)}$ and $g^{(i)}$ is converted into $g^{(i+1)}$ in the ith run for $1 \leq i \leq N$.

Termination: Let

$$A = \{p \text{ is a Boolean function} \mid p \prec h\}.$$

It is clear that A is a finite set. Since, in the while-loop, we have that

$$h^{(i+1)} \prec h^{(i)} \text{ for } 1 \leq i \leq N,$$

the while-loop must terminate within $|A|$ steps. Thus $N \leq |A|$.

Correctness: First, if $h = f * g$ for some Boolean function g, then it is clear that the algorithm will output g by the discussions before this lemma. Note that the order of solving $g_{[1]}, g_{[2]}, \ldots, g_{[d]}$ used in Table 3 is:

$$g_{[j_1]}, g_{[j_2]}, \ldots, g_{[j_a]}, \mathrm{HT}(g_{[j_1]}) \preceq \mathrm{HT}(g_{[j_2]}) \preceq \cdots \preceq \mathrm{HT}(g_{[j_a]})$$

where $\{g_{[j_1]}, g_{[j_2]}, \ldots, g_{[j_a]}\} = \{g_{[j]} \mid g_{[j]} \neq 0,\ 1 \leq j \leq d\}$.

Second, we show that if $h^{(N+1)} = 0$ when the while-loop terminate, then $h = f * g^{(N+1)}$. We conclude that

$$f * g^{(i)} = h \oplus h^{(i)} \text{ for } 1 \leq i \leq N+1. \tag{8}$$

It is trivially true for $i = 1$ since $g^{(1)} = 0$ and $h^{(1)} = h$. Now suppose (8∗) is true for) is }∗(gTdf(=)]T9 true forloa

Therefore (8) holds for the $i+1$. It immediately follows from (8) that if $h^{(N+1)} = 0$, then $h = f * g^{(N+1)}$, and so $g^{(N+1)}$ is the desirable Boolean function. ■

```
**Specification:** $v = (v(1), v(2)) \longleftarrow$ FIND-RIGHT-FACTOR$(h, f)$
**Given:** a Boolean function $h \in \mathcal{C}^*$ and a linear Boolean function $f$
**Find:** a Boolean function $g \in \mathcal{C}^*$ such that $h = f * g$ if $g$ exists
begin
$g \longleftarrow 0$
$n \longleftarrow \mathrm{ord}(f)$
while $h \neq 0$ do
    $d \longleftarrow \deg(h)$
    $t \longleftarrow \min\{\mathrm{HT}(h_{[l]}) \mid h_{[l]} \neq 0, 1 \leq l \leq d\}$
   (assume $\deg(t) = k$ and $t = x_{j_1}x_{j_2}\cdots x_{j_k}$)
    if $i_1 \geq n$ then
        $g \longleftarrow g \oplus x_{j_1-n}x_{j_2-n}\cdots x_{j_k-n}$
        $h \longleftarrow h \oplus (f * (x_{j_1-n}x_{j_2-n}\cdots x_{j_k-n}))$
    else return(False, 0)
    end if
end while
return(True, $g$)
```

Table 3. Algorithm FIND-RIGHT-FACTOR

Remark 10 *Let us denote by $\preceq_d$ the term order which first compares total degrees and then breaks ties by the inverse lexicographical order. The strategy of the selection of the term t in Table 2 during executions of the while-loop can be replaced by*

$$t_1 \longleftarrow \mathrm{HT}(h) \ \textit{w.r.t.} \ \preceq \ \textit{or} \ \ t_2 \longleftarrow \mathrm{HT}(h) \ \textit{w.r.t.} \ \preceq_d .$$

However, for a false Boolean function f, i.e., $h \neq f * g$ for any Boolean function g, the strategy used in the algorithm may yield a faster termination of the while-loop since the selected term t satisfies that $t \preceq t_1$ and $t \preceq t_2$.

Remark 11 *Usually, during the algorithm, $|T(h)|$ decreases for a right f, while $|T(h)|$ increases for a false f. Generally speaking, $|T(g)|$ is less than $|T(h)|$. Hence, when the run time is too long, setting an upper bound for the number of rounds (say, $|T(h)|$) of the while-loop will not affect the results of the algorithm in general and make the algorithm terminate faster. Similar principle can be used for the algorithm of Table 5.*

Next, we show how to obtain linear functions f such that $f \parallel_L h$. Let $d \in \mathbb{N}^*$. For a homogeneous Boolean function

$$h = \bigoplus_{j=1}^{N} x_{i_{j,1}} x_{i_{j,2}} \cdots x_{i_{j,d}},\ i_{j,1} < i_{j,2} < \cdots < i_{j,d},$$

of degree d, define $\Phi(h)$ to be the gcd of the following d polynomials determined by h in $\mathbf{F}_2[x]$:

$$p_{h,k}(x) = \phi(x_{i_{1,k}} \oplus x_{i_{2,k}} \oplus \cdots \oplus x_{i_{N,k,}}), 1 \leq k \leq d, \tag{9}$$

where ϕ is defined by (2), i.e.,

$$\Phi(h) = \gcd(p_{h,1}(x), p_{h,2}(x), \ldots, p_{h,d}(x)) \in \mathbf{F}_2[x].$$

Furthermore, based on this notation, for any Boolean function h of degree d without a constant term, define

$$\Phi^*(h) = \gcd(\Phi(h_{[1]}), \Phi(h_{[2]}), \ldots, \Phi(h_{[d]})) \in \mathbf{F}_2[x].$$

Lemma 12 *Let $h, f, g \in \mathcal{C}^*$ such that $h = f * g$. If f is linear, then $\phi(f)$ divides $\Phi^*(h)$ in $\mathbf{F}_2[x]$.*

Proof. Let $\deg(h) = d$. Clearly it suffices to prove that $\phi(f)$ divides $\Phi(h_{[i]})$ for $i = 1, 2, \ldots, d$. By Lemma 6 $h_{[i]} = f * g_{[i]}$ for $i = 1, 2, \ldots, d$. Thus without loss of generality, we assume that both h and g are homogeneous of degree d.

Let

$$g = \bigoplus_{j=1}^{N} x_{i_{j,1}} x_{i_{j,2}} \cdots x_{i_{j,d}} \text{ and } f = c_0 x_0 \oplus c_1 x_1 \oplus \cdots \oplus c_n x_n.$$

Then

$$h = f * g = \bigoplus_{l=0}^{n} \bigoplus_{j=1}^{N} c_l x_{i_{j,1}+l} x_{i_{j,2}+l} \cdots x_{i_{j,d}+l}.$$

It follows that for $k = 1, 2, \ldots, d$,

$$p_{h,k}(x) = \phi\left(\bigoplus_{l=0}^{n} \bigoplus_{j=1}^{N} c_l x_{i_{j,k}+l}\right) = \phi\left(f * \left(\bigoplus_{j=1}^{N} x_{i_{j,k}}\right)\right) = \phi(f) p_{g,k}(x),$$

where $p_{h,k}(x)$ and $p_{g,k}(x)$ are defined by (9). This implies that $\phi(f)$ divides $p_{h,k}(x)$ in $\mathbf{F}_2[x]$ for $k = 1, 2, \ldots, d$, and so $\phi(f)$ divides $\Phi(h)$. ■

Lemma 12 implies that by factoring $\Phi^*(h)$ in $\mathbf{F}_2[x]$, we can get all the linear functions f such that $f \parallel_L h$. Then the following theorem immediately follows from Theorem 9 and Lemma 12.

Theorem 13 *Let $h \in \mathcal{C}^*$. Then the algorithm* EQUAL-DEGREE *of Table* 4 *computes all Boolean function pairs $(f, g) \in \mathcal{C}^* \times \mathcal{C}^*$ such that $h = f * g$ and* $\deg(f) = 1$.

```
Specification: $S \longleftarrow$ EQUAL-DEGREE($h$)
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 $h = f * g$ and $f$ is linear
begin
$S \longleftarrow \varnothing$
$\Omega \longleftarrow \{\phi^{-1}(a(x)) \mid a(x)$ is a divisor of $\Phi^*(h)$ in $\mathbf{F}_2[x]$ $\}$
for all $f \in \Omega$ do
    $v \longleftarrow$ FIND-RIGHT-FACTOR($h, f$)
    if $v(1) =$ True then
        $S \longleftarrow S \cup \{(f, v(2))\}$
    end if
end for
return($S$)
```

Table 4. Algorithm EQUAL-DEGREE

3.3 Find the unique left $*$-factor corresponding to a given right $*$-factor

Given $h \in \mathcal{C}^*$ with $\deg(h) = d$ and a right $*$-factor g of h. In this subsection, we discuss how to find a Boolean function f such that $h = f * g$. By Corollary 5, we know that the function f is unique.

First, we consider the case $\deg(g) = \deg(h)$. In this case, we have that f is linear by Lemma 8. Since by Lemma 6

$$h_{[i]} = f * g_{[i]} \text{ for } 1 \leq i \leq d,$$

it follows that if

$$\mathrm{HT}(g_{[k]}) = \min\{\mathrm{HT}(g_{[1]}), \mathrm{HT}(g_{[2]}), \ldots, \mathrm{HT}(g_{[d]})\}, 1 \leq k \leq d,$$

then

$$\begin{aligned} \mathrm{HT}(h_{[k]}) &= \min\{\mathrm{HT}(h_{[1]}), \mathrm{HT}(h_{[2]}), \ldots, \mathrm{HT}(h_{[d]})\} \\ &= \mathrm{HT}(f) * \min\{\mathrm{HT}(g_{[1]}), \mathrm{HT}(g_{[2]}), \ldots, \mathrm{HT}(g_{[d]})\}. \end{aligned} \tag{10}$$

Let us write

$$\mathrm{HT}(g_{[k]}) = x_{i_1} x_{i_2} \cdots x_{i_k} \text{ and } \mathrm{HT}(h_{[k]}) = x_{j_1} x_{j_2} \cdots x_{j_k},$$

where $i_1 < i_2 < \cdots < i_k$ and $j_1 < j_2 < \cdots < j_k$. Then (10) implies that

$$\mathrm{HT}(f) = x_{j_1 - i_1} = x_{j_2 - i_2} = \cdots = x_{i_k - j_k}.$$

Similarly, we can solve $\mathrm{HT}(f \oplus \mathrm{HT}(f))$, since

$$h_{[k]} \oplus \mathrm{HT}(f) * g_{[k]} = (f \oplus \mathrm{HT}(f)) * g_{[k]}.$$

Based on this observation we give the following theorem.

Theorem 14 *Let* $h, g \in \mathcal{C}^*$ *with* $\deg(g) = \deg(h)$. *Then the algorithm* FIND-LEFT-LINEAR *of Table* 5 *computes the unique linear Boolean function* f *such that* $h = f * g$ *if* f *exists.*

```
Specification: $v = (v(1), v(2)) \longleftarrow$ FIND-LEFT-LINEAR($h, g$)
Given: two Boolean functions $h, g \in \mathcal{C}^*$ with $\deg(g) = \deg(h)$
Find: a linear Boolean function $f$ such that $h = f * g$ if $f$ exists
begin
$s \longleftarrow \mathrm{HT}(g_{[k]}) = \min\{\mathrm{HT}(g_{[1]}), \mathrm{HT}(g_{[2]}), \ldots, \mathrm{HT}(g_{[d]})\}$
        (assume $s = x_{i_1} x_{i_2} \cdots x_{i_k}$)
$\eta \longleftarrow h_{[l]}$ where $\mathrm{HT}(h_{[l]}) = \min\{\mathrm{HT}(h_{[1]}), \mathrm{HT}(h_{[2]}), \ldots, \mathrm{HT}(h_{[d]})\}$
$f \longleftarrow 0$
if $k = l$ then
    while $\eta \neq 0$ do
        $t \longleftarrow \mathrm{HT}(\eta)$ (assume $t = x_{j_1} x_{j_2} \cdots x_{j_k}$)
        $\varepsilon_u \longleftarrow j_u - i_u$ for $u = 1, 2, \ldots, k$
        if $\varepsilon_1 \geq 0$ and $\varepsilon_1 = \varepsilon_2 = \cdots = \varepsilon_k$ then
            $\eta \longleftarrow \eta \oplus x_{\varepsilon_1} * g_{[k]}$
            $f \longleftarrow f \oplus x_{\varepsilon_1}$
        else return(False, 0)
        end if
    end while
else return(False, 0)
end if
if $h = f * g$ then return(True, $f$)
else return(False, 0)
end if
```

Table 5. Algorithm FIND-LEFT-LINEAR

Proof. *Termination:* Since the term order of $\mathrm{HT}(\eta)$ in the while-loop strictly decreases, the while-loop will terminate.

Correctness: By the last if-condition, it is clear that if the algorithm output f, then $h = f * g$. On the other hand, if $h = f * g$ for some Boolean function f, then it follows from

Corollary 5 and Lemma 8 that f is unique and linear. Finally, it follows from the above discussions that the algorithm will output f. ■

For a Boolean function f with $\deg(\mathrm{HT}(f)) = k > 1$ and $\mathrm{HT}(f) = x_{i_1}x_{i_2}\cdots x_{i_k}$, it is clear that we can write

$$f = x_{i_2}\cdots x_{i_k}\cdot p \oplus q$$

where p, q are Boolean functions such that $\mathrm{HT}(p) = x_{i_1}$ and every term of q is not divisible by $x_{i_2}x_{i_3}\cdots x_{i_k}$. We denote the Boolean function p by $\Gamma(f)$. Note that the degree of $\Gamma(f)$ may be greater than 1. For example, if $f = x_4x_5x_6 \oplus x_1x_2x_5x_6 \oplus x_1x_5x_6 \oplus x_2x_6 \oplus x_2x_4x_5$, then $\mathrm{HT}(f) = x_4x_5x_6$ and $\Gamma(f) = x_4 \oplus x_1x_2$.

Lemma 15 *Let $g \in \mathcal{C}^*$ with $\mathrm{ord}(g) = m$, and let h, f be two Boolean functions which are not constants. If $h = f * g$ and $\deg(\mathrm{HT}(h)) > 1$, then $\Gamma(h) = \Gamma(f) * g$.*

Proof. Let $\mathrm{HT}(f) = x_{i_1}x_{i_2}\cdots x_{i_k}$ where $k \geq 1$. Then by Corollary 4

$$\mathrm{HT}(h) = x_{i_1+m}x_{i_2+m}\cdots x_{i_k+m}, \tag{11}$$

and so $k > 1$. Note that f can be written

$$f = \Gamma(f)\cdot(x_{i_2}x_{i_3}\cdots x_{i_k}) \oplus q$$

where $t \prec x_{i_2}x_{i_3}\cdots x_{i_k}$ for all $t \in T(q)$. Then

$$h = f * g = (\Gamma(f) * g)\cdot((x_{i_2}x_{i_3}\cdots x_{i_k}) * g) \oplus q * g. \tag{12}$$

Since

$$\mathrm{HT}(\Gamma(f)) = x_{i_1} \prec x_{i_2} \quad \text{and} \quad \mathrm{HT}(q) \prec x_{i_2}x_{i_3}\cdots x_{i_k},$$

it follows from Corollary 4 that

$$\mathrm{HT}(\Gamma(f) * g) \prec x_{i_2} * x_m = x_{i_2+m} \tag{13}$$

and

$$\mathrm{HT}(q * g) \prec (x_{i_2}x_{i_3}\cdots x_{i_k}) * x_m = x_{i_2+m}x_{i_3+m}\cdots x_{i_k+m}. \tag{14}$$

On the other hand, by Corollary 4

$$\mathrm{HT}((x_{i_2}x_{i_3}\cdots x_{i_k})*g) = x_{i_2+m}x_{i_3+m}\cdots x_{i_k+m}. \tag{15}$$

Then (13) and (15) imply that

$$(\Gamma(f)*g)\cdot((x_{i_2}x_{i_3}\cdots x_{i_k})*g) = (\Gamma(f)*g)\cdot x_{i_2+m}x_{i_3+m}\cdots x_{i_k+m}\oplus p,$$

where no term in p is divisible by $x_{i_2+m}x_{i_3+m}\cdots x_{i_k+m}$, and (14) implies that no term in $q*g$ is divisible by $x_{i_2+m}x_{i_3+m}\cdots x_{i_k+m}$. Hence it can be seen from (12) that $\Gamma(h)=\Gamma(f)*g$. ■

Theorem 16 *Let* $h, g \in \mathcal{C}^*$ *with* $\deg(h) \geq \deg(g)$. *Then the algorithm* FIND-LEFT-NONLINEAR *of Table* 6 *(see page* 28*) computes the unique Boolean function* f *such that* $h = f*g$ *if* f *exists.*

Proof. Termination: Since the term order of $\mathrm{HT}(h)$ strictly decreases, the while-loop will terminate.

Correctness: If $\deg(h)=\deg(g)$, then the assertion follows from Theorem 14. Thus we need only to consider the case $\deg(h)>\deg(g)$.

Suppose there are N runs through the while-loop, where $h^{(i)} \to h^{(i+1)}$ and $f^{(i)} \to f^{(i+1)}$ for $0 \leq i < N$. Similar with the proof of Theorem 9, it can be shown that

$$f^{(i)}*g = h\oplus h^{(i)}, 0\leq i\leq N. \tag{16}$$

Hence if $h^{(N)}=0$, then $h=f^{(N)}*g$.

Next, we show if $h=f*g$ for some Boolean function f, then the while-loop will terminate with $h^{(N)}=0$. For $0\leq i\leq N-1$, taking $h=f*g$ into (16) yields

$$\left(f^{(i)}\oplus f\right)*g = h^{(i)}.$$

This shows that i_1 in Table 6 is not less than m. Furthermore, if $\deg(\mathrm{HT}(h^{(i)})) > 1$, then by Lemma 15

$$\Gamma(h^{(i)}) = \Gamma\left(f^{(i)} \oplus f\right) * g,$$

which implies that $v(1)$ is always true if FIND-LEFT-LINEAR($\Gamma(h), g$) is executed. Thus one of the three if-conditions in the while loop is satisfied for each round until $h^{(N)} = 0$ is attained.

Finally the uniqueness follows from Corollary 5. ■

3.4 Find all right $*$-factors of a given degree

Given $h \in \mathcal{C}^*$ and an integer d, in this subsection, we shall show how to obtain all Boolean function pairs $(f, g) \in \mathcal{C}^* \times \mathcal{C}^*$ such that $h = f * g$ with $\deg(g) = d$. It follows from Lemma 8 that we need only to consider $d \leq \deg(h)$. Moreover, in Subsection 3.2, we have solved the case $d = \deg(h)$. Thus the main aim of this subsection is to solve the case $d < \deg(h)$.

We first introduce some notations. Let f be a Boolean function which is not a constant. For any given term $s = x_{i_1}x_{i_2}\cdots x_{i_k}$, where $k \geq 1$, let us denote

$$\langle \frac{f}{s} \rangle = \bigoplus_{t \prec x_{i_1} \text{ and } t \cdot s \in T(f)} t,$$

which is 0 if no such term t exists. For any positive integer $j \leq \deg(f)$, denote

$$f_{\geq[j]} = \bigoplus_{i=j}^{\deg(f)} f_{[i]}.$$

Based on the above two notations, for any positive integer d, denote

$$\Delta_d(f) = \begin{cases} \langle \frac{f_{\geq[2]}}{x_n} \rangle, \text{ if } \deg(f) \geq d; \\ f, \text{ otherwise,} \end{cases}$$

where $n = \mathrm{ord}(f_{\geq[d]})$. Furthermore, for any $e \in \mathbb{N}^*$, define $\Delta_d^e(f) = \Delta_d\left(\Delta_d^{e-1}(f)\right)$ where $\Delta_d^0(f) = f$, composition of the function Δ_d. In particular, if e is the first nonnegative integer such that $\deg(\Delta_d^e(f)) < d$, then define $\Delta_d^*(f) = \Delta_d^e(f)$.

Example 17 *Let* $f = x_3x_6x_7 \oplus x_4x_5x_6 \oplus x_1x_3x_4x_6 \oplus x_2x_4x_6$*. Then*

$$\langle \frac{f}{x_4x_6} \rangle = x_1x_3 \oplus x_2 \quad \textit{and} \ \langle \frac{f}{x_3x_6} \rangle = 0.$$

Example 18 *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*

$$\Delta_3(f) = \langle \frac{f_{\geq[2]}}{x_8} \rangle = x_6x_7 \oplus x_4x_5x_6 \quad \textit{and} \quad \Delta_3^2(f) = \langle \frac{f_{\geq[2]}}{x_6x_8} \rangle = x_4x_5.$$

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

Lemma 19 *Let* h, f, g *be three Boolean functions such that* g *is nonsingular and* $h = f * g$*. If* $\deg(h) > \deg(g)$*, then*

$$\Delta_{d+1}(h) = \langle \frac{f_{[1]} * g_{\geq[2]}}{x_r} \rangle \oplus \Delta_2(f) * g, \tag{17}$$

where $d = \deg(g)$ *and* $r = \mathrm{ord}(h_{\geq[d+1]})$*.*

Proof. Let $\mathrm{ord}(f_{\geq[2]}) = n$ and $\mathrm{ord}(g) = m$. Since $\deg(h) > \deg(g)$, it follows from Lemma 8 that $\deg(f) > 1$. Then f can be written

$$f = f_{[1]} \oplus f_{\geq[2]} = f_{[1]} \oplus \Delta_2(f) \cdot x_n \oplus q,$$

where $t \prec x_n$ for all $t \in T(q)$. Thus

$$h = f * g = f_{[1]} * g \oplus (\Delta_2(f) * g) \cdot (x_n * g) \oplus q * g. \tag{18}$$

Since by Lemma 8 $\deg(f_{[1]} * g) = d$, it can be seen that

$$\mathrm{ord}\left(h_{\geq[d+1]}\right) = \mathrm{ord}\left(((\Delta_2(f) * g) \cdot (x_n * g) \oplus q * g)_{\geq[d+1]}\right).$$

On one hand, since by Lemma 8 $\deg(\Delta_2(f) * g) \geq \deg(g) = d$ and

$$\mathrm{HT}(\Delta_2(f) * g) \prec \mathrm{HT}(x_n * g) = x_{n+m},$$

we have that

$$\mathrm{ord}(p_{\geq[d+1]}) = n + m \ \text{ and } \ \Delta_{d+1}(p) = \Delta_2(f) * g, \tag{19}$$

where $p = (\Delta_2(f) * g) \cdot (x_n * g)$. On the other hand, since $\mathrm{HT}(q) \prec x_n$, we have that

$$\mathrm{HT}(q * g) \prec x_{n+m}. \tag{20}$$

Hence

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

and

$$\Delta_{d+1}(h) = \langle \frac{f_{[1]} * g_{\geq[2]}}{x_{n+m}} \rangle \oplus \Delta_2(f) * g.$$

This completes the proof. ■

We note that the Boolean function

$$p = \langle \frac{f_{[1]} * g_{\geq[2]}}{x_r} \rangle$$

appearing on the right hand side of (17) satisfies: (1) $\deg(p) < \deg(g)$; (2) $p = 0$ if $\deg(g) = 1$. These observations lead to the following corollary.

Corollary 20 *Let h, f, g be as described in Lemma 19. If g is linear, then*

$$\Delta_2(h) = \Delta_2(f) * g.$$

Remark 21 *Corollary 20 implies that*

$$\Delta_2^*(h) = \Delta_2^*(f) * g.$$

Since $\Delta_2^*(h), \Delta_2^*(f), g$ are linear, we have that

$$\phi(\Delta_2^*(h)) = \phi(\Delta_2^*(f)) \cdot \phi(g)$$

in $\mathbf{F}_2[x]$. Thus, we can find g by factor $\phi(\Delta_2^*(h))$ in $\mathbf{F}_2[x]$. This result is distinct from Theorem 1 in [4], though they look similar.

Corollary 22 *Let h, f, g be as described in Lemma 19. Then*

$$\left(\Delta_{d+1}^*(h)\right)_{[d]} = \Delta_2^*(f) * g_{[d]}$$

where $d = \deg(g)$.

Theorem 23 *Let $h \in \mathcal{C}^*$ and $d \in \mathbb{N}^*$ with $d \leq \deg(h)$. Then the algorithm* GIVENDEG *of Table* 7 *(see page* 29*) together with the algorithm* SUBALG-FOR-GIVENDEG *of Table* 8 *(see page* 30*) computes all Boolean function pairs $(f, g) \in \mathcal{C}^* \times \mathcal{C}^*$ such that $h = f * g$ and* $\deg(g) = d$.

Proof. Termination: It is trivially true that the two algorithms will terminate.

Correctness: If the algorithm of Table 7 outputs S, then the first if-condition of Table 8 implies that $h = f * g$ for all $(f, g) \in S$. In the following, we need only to show that if $h = f * g$ for some Boolean functions f and g with $\deg(g) = d$, then $(f, g) \in S$.

If $d = \deg(h)$, then $S =$ EQUAL-DEGREE(h), and so $(f, g) \in S$ by Theorem 13. Thus we assume that $d < \deg(h)$.

Let k be the minimal integer such that $\Delta^*_{d+1}(h) = \Delta^k_{d+1}(h)$. Denote

$$h^{(i)} = \Delta^i_{d+1}(h) \text{ and } f^{(i)} = \Delta^i_2(f) \text{ for } i = 0, 1, \ldots, k,$$

and

$$\operatorname{ord}(h) = m_0 \text{ and } \operatorname{ord}(h^{(i)}_{\geq[d+1]}) = m_{i+1} \text{ for } i = 0, 1, \ldots, k-1.$$

It is clear that $m_0 > m_1 > \cdots > m_k$.

First, it can be seen that the while-loop in Table 7 computes $\Delta^*_{d+1}(h) = h^{(k)}$. By Corollary 22 we have that

$$h^{(k)}_{[d]} = f^{(k)} * g_{[d]}.$$

Since $f^{(k)}$ is linear, it can be seen that $(f^{(k)}, g_{[d]}) \in S_1$.

Second, if $\deg(g) = 1$, then $g_{[d]} = g$. Otherwise, we claim that

$$h^{(k)} \oplus p = f^{(k)} * g \tag{21}$$

for some Boolean function p of degree less than d. Moreover, the Boolean function p satisfies that for $j = 1, 2, \ldots, d-1$,

$$\begin{aligned} T(p_{[d-j]}) \quad \subseteq \quad & \{t_p \text{ is a term of degree } d-j \mid \text{there are an integer } 1 \leq u \leq \min\{k, j\}, \\ & \text{an integer } m_{k-u+1} - \widetilde{\delta}_g < a < m_{k-u} - \widehat{\delta}_g, \text{and a term } t \in T(g_{[d-j+u]}) \\ & \text{such that } x_a * t = t_p x_{m_k} \cdots x_{m_{k-u+2}} x_{m_{k-u+1}}\}, \qquad (22) \end{aligned}$$

where $\widetilde{\delta}_g = m_k - 1 - \text{ord}(f)$ and $\widehat{\delta}_g = \text{ord}(g_{[d]}) + 1$, by which it can be seen that $p_{[d-j]}$ is only related with $g_{[d-j+1]}, g_{[d-j+2]}, \ldots, g_{[d]}$. In the following we shall prove (21) and (22).

Recursively using Lemma 19 yields

$$\begin{aligned} h^{(1)} \oplus \langle \frac{p^{(1)}}{x_{m_1}} \rangle \quad &= \quad f^{(1)} * g, \text{ where } p^{(1)} = f^{(0)}_{[1]} * g_{\geq [2]}, \\ h^{(2)} \oplus \langle \frac{p^{(1)}}{x_{m_2} x_{m_1}} \rangle \oplus \langle \frac{p^{(2)}}{x_{m_2}} \rangle \quad &= \quad f^{(2)} * g, \text{ where } p^{(2)} = f^{(1)}_{[1]} * g_{\geq [2]}, \\ &\vdots \\ h^{(k)} \oplus \left(\bigoplus_{i=1}^{k} \langle \frac{p^{(i)}}{x_{m_k} \cdots x_{m_{i+1}} x_{m_i}} \rangle \right) \quad &= \quad f^{(k)} * g, \text{ where } p^{(k)} = f^{(k-1)}_{[1]} * g_{\geq [2]}. \qquad (23) \end{aligned}$$

This shows that (21) holds and

$$p = \bigoplus_{i=1}^{k} \langle \frac{p^{(i)}}{x_{m_k} \cdots x_{m_{i+1}} x_{m_i}} \rangle. \qquad (24)$$

If $t_p \in T(p)$ and $\deg(t_p) = d - j$, then it follows from (24) that

$$t_p \in T\left(\langle \frac{p^{(k-u+1)}}{x_{m_k} \cdots x_{m_{k-u+2}} x_{m_{k-u+1}}} \rangle \right) \quad \text{for some } 1 \leq u \leq \min\{k, j\},$$

and so

$$t_p x_{m_k} \cdots x_{m_{k-u+2}} x_{m_{k-u+1}} \in T(p^{(k-u+1)}) = T\left(f^{(k-u)}_{[1]} * g_{\geq [2]} \right).$$

Thus there exist $x_a \in T(f^{(k-u)}_{[1]})$ and $t \in T(g_{\geq [2]})$ such that

$$t_p x_{m_k} \cdots x_{m_{k-u+2}} x_{m_{k-u+1}} = x_a * t. \qquad (25)$$

It follows that

$$\deg(t) = \deg(t_p) + u = d - j + u \qquad (26)$$

and

$$m_{k-u+1} - \mathrm{ord}(g) < m_{k-u+1} - \mathrm{ord}(t) = a \leq \mathrm{ord}(f^{(k-u)}_{[1]}), \tag{27}$$

On one hand, since by (23)

$$\begin{aligned} \mathrm{ord}(g) &= \mathrm{ord}\left(h^{(k)} \oplus \left(\bigoplus_{i=1}^{k} \langle \frac{p^{(i)}}{x_{m_k} \cdots x_{m_{i+1}} x_{m_i}} \rangle \right)\right) - \mathrm{ord}(f^{(k)}) \\ &\leq (m_k - 1) - \mathrm{ord}(f^{(k)}) = \widetilde{\delta}_g, \end{aligned}$$

it follows that

$$m_{k-u+1} - \mathrm{ord}(g) \geq m_{k-u+1} - \widetilde{\delta}_g. \tag{28}$$

On the other hand, since

$$\begin{aligned} \mathrm{ord}(f^{(k-u)}_{[1]}) &\leq \mathrm{ord}(f^{(k-u)}) \\ &< \mathrm{ord}(f^{(k-u-1)}) \\ &\leq \mathrm{ord}(h^{(k-u-1)}) - \mathrm{ord}(g) \\ &\leq \mathrm{ord}(h^{(k-u-1)}) - (\mathrm{ord}(g_{[d]}) + 1), \end{aligned}$$

it follows that

$$\mathrm{ord}(f^{(k-u)}_{[1]}) < m_{k-u} - \widehat{\delta}_g. \tag{29}$$

Taking (28) and (29) into (27) leads to

$$m_{k-u+1} - \widetilde{\delta}_g < a < m_{k-u} - \widehat{\delta}_g. \tag{30}$$

Thus it follows from (25), (26) and (30) that (22) holds.

Third, based on (21) and (22), we prove that SUBALG-FOR-GIVENDEG with last input given by $1 \leq j \leq d-1$ will compute $g_{[d-j]}$. We remark that j runs from 1 to $d-1$.

Note that $g_{[d]}$ has been solved. Suppose that $g_{[d]}, g_{[d-1]}, \ldots, g_{[d-j+1]}$ have been solved. It can be seen that the first for-loop constructs the set (22)(see Ω in Table 8), and the second for-loop computes all possible $p_{[d-j]}$ (see $\bigoplus_{t \in A} t$ in Table 8). Since $f^{(k)}$ is linear, (21) implies that

$$h^{(k)}_{[d-j]} \oplus p_{[d-j]} = f^{(k)} * g_{[d-j]}.$$

Thus, $g_{[d-j]}$ can be solved by FIND-RIGHT-FACTOR when the correct $p_{[d-j]}$ is found.

Finally, since one of the branches of the recursive algorithm SUBALG-FOR-GIVENDEG will yield g, the first if-condition implies that $(f, g) \in S$. ■

4 Conclusions

In this paper we give a series of algorithms to decompose an NFSR into a cascade connection of two smaller NFSRs. Although for the worst case, the computation complexity is exponential in the stage of the NFSR, they are shown to be efficient in practice. By our algorithms, we completed the search of all possible decompositions in $\mathcal{C}^*$ of the 160-stage NFSR used in Grain, a cascade connection of a 80-stage LFSR into a 80-stage NFSR, in about 10 minutes on a single PC, and found that it has no other decomposition. Besides, the 256-stage NFSR used in Grain-128 also has no other decomposition in $\mathcal{C}^*$.

```
**Specification:** $v = (v(1), v(2)) \longleftarrow$ FIND-LEFT-NONLINEAR$(h, g)$
**Given:** Two Boolean functions $h, g \in \mathcal{C}^*$ with $\deg(h) \geq \deg(g)$
**Find:** a Boolean function $f$ such that $h = f * g$ if $f$ exists
begin
if $\deg(h) = \deg(g)$ then
  $v \longleftarrow$ FIND-LEFT-LINEAR$(h, g)$
  return$(v)$
end if
$f \longleftarrow 0$
$m \longleftarrow \mathrm{ord}(g)$
while $h \neq 0$ do
  $t \longleftarrow \mathrm{HT}(h)$  (assume $t = x_{i_1} x_{i_2} \cdots x_{i_k}$)
  if $i_1 \geq m$ and $\deg(t) = 1$ then
    $h \longleftarrow h \oplus (x_{i_1 - m} * g)$
    $f \longleftarrow f \oplus x_{i_1 - m}$
  else if $i_1 \geq m$ and $\deg(\Gamma(h)) > \deg(g)$ then
    $h \longleftarrow h \oplus ((x_{i_1-m} x_{i_2-m} \cdots x_{i_k-m}) * g)$
    $f \longleftarrow f \oplus x_{i_1-m} x_{i_2-m} \cdots x_{i_k-m}$
  else if $i_1 \geq m$ and $\deg(\Gamma(h)) = \deg(g)$ then
    $v \longleftarrow$ FIND-LEFT-LINEAR$(\Gamma(h), g)$
    if $v(1) =$ True then
      $f' \longleftarrow v(2) \cdot (x_{i_2-m} \cdots x_{i_k-m})$
      $h \longleftarrow h \oplus (f' * g)$
      $f \longleftarrow f \oplus f'$
    else return(False, 0)
    end if
  else return(False, 0)
  end if
end while
return(True, $f$)
```

Table 6. Algorithm FIND-LEFT-NONLINEAR

```
Specification: $S \longleftarrow$ GIVENDEG$(h, d)$
Given: a Boolean function $h \in \mathcal{C}^*$ and a positive integer $d \leq \deg(h)$
Find: a finite set $S$ of Boolean function pairs $(f, g) \in \mathcal{C}^* \times \mathcal{C}^*$ such
     that $h = f * g$ and $\deg(g) = d$
begin
$S \longleftarrow \varnothing$
if $\deg(h) = d$ then
  $S \longleftarrow$ EQUAL-DEGREE$(h)$
else
  $k \longleftarrow 0$
  $M \longleftarrow \{\mathrm{ord}(h) + 1\}$
  $\eta \longleftarrow h$
  while $\deg(\eta) > d$ do
     $k \longleftarrow k + 1$
     $M \longleftarrow M \cup \{\mathrm{ord}(\eta_{\geq[d+1]})\}$
     $\eta \longleftarrow \Delta_{d+1}(\eta)$
  end while
  if $\deg(\eta) = d$ then
     $S_1 \longleftarrow$ EQUAL-DEGREE$(\eta_{[d]})$
     for all $(f, g) \in S_1$ do
        SUBALG-FOR-GIVENDEG$(\eta, h, f, g, S, k, d, M, 1)$
     end for
  end if
end if
return$(S)$
```

Table 7. Algorithm GIVENDEG

```
Specification: SUBALG-FOR-GIVENDEG($\eta, h, f, g, S, k, d, M, j$)
               (a recursive algorithm)
Given: Boolean functions $\eta, h, f, g$; a finite set $S$ of Boolean functions;
       $k, d, j \in \mathbb{N}^*$; a list of positive integers $M = \{m_0, m_1, \ldots, m_k\}$
       where $m_0 > m_1 > \cdots > m_k$
Find: no output but all desirable results are added to the set $S$
begin
$\Omega \longleftarrow \varnothing$
$\widetilde{\delta}_g \longleftarrow m_k - 1 - \mathrm{ord}(f)$
$\widehat{\delta}_g \longleftarrow \mathrm{ord}(g_{[d]}) + 1$
if $j = d$ then
    $v$ = FIND-LEFT-NONLINEAR($h, g$)
    if $v(1)$ = True then
        $S \longleftarrow S \cup \{(v(2), g)\}$
    end if
else
    for $u$ from 1 to $\min\{j, k\}$ do
        $\Omega \longleftarrow \Omega \cup \{t_p$ is a term $\mid \deg(t_p) = d - j$ and $x_a * t = t_p x_{m_k} \cdots x_{m_{k-u+2}} x_{m_{k-u+1}}$
                    for some term $t \in T(g_{[d-j+u]})$ and some integer
                    $m_{k-u+1} - \widetilde{\delta}_g < a < m_{k-u} - \widehat{\delta}_g\}$
    end for
    for all $A \subseteq \Omega$ do   (include $A = \varnothing$)
        $\eta' \longleftarrow \eta \oplus \left(\bigoplus_{t \in A} t\right)$
        $v \longleftarrow$ FIND-RIGHT-FACTOR($\eta'_{[d-j]}, f$)
        if $v(1)$ = True then
            $g \longleftarrow g \oplus v(2)$
            SUBALG-FOR-GIVENDEG($\eta, h, f, g, S, k, d, M, j + 1$)
        end if
    end for
end if
```

Table 8. Algorithm SUBALG-FOR-GIVENDEG

References

[1] Hell M., Johansson T., Meier W.: The Grain family of stream ciphers. In: New Stream Cipher Designs: The eSTREAM Finalists. Lecture Notes in Computer Science, **4986**, pp. 179–190. Springer-Verlag, New York (2008).

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

[3] Green D.H., Dimond K.R.: Nonlinear product-feedback shift registers. PROC. IEE **117**(4), pp. 681–686 (1970).

[4] Ma Z., Qi W.F.: On the decomposition of an NFSR into the cascade connection of an NFSR into an LFSR. To appear in J. Complex. DOI: 10.1016/j.jco.2012.09.003.

[5] Golomb S.W.: Shift Register Sequences, Aegean Park Press, California (1982).

[6] Mykkeltveit J., Siu M.K., Tong P.: On the cycle structure of some nonlinear shift register sequences. Information and Control **43**, 202–215 (1979).