Attack based on Direct Sum Decomposition against the Nonlinear Filter Generator

Abstract. The nonlinear filter generator (NLFG) is a common building block in many stream ciphers. In this paper we present the direct sum decomposition of the NLFG output sequence that leads to a linear equation system in the initial state of the NLFG and to a powerful algebraic attack. We show that the coefficients of the equation system rely only on the NLFG structure and thus can be determined in an off-line manner, speeding up the attack. Like the known Rønjom-Helleseth attack, this attack requires an on-line computation with complexity $O(LC)$ (LC is the linear complexity of the output sequence); but totally it requires far fewer output bits and significantly less matrix computation. Different from most prior work, this paper carefully studies the success probability of the attack. It is shown that the probability of this algebraic attack is $1-2^{-\phi(2^n-1)}$ ($\phi(\cdot)$ is the Euler function), which is much greater than the success probability $1-2^{-n}$ of the Rønjom-Helleseth attack.

Keywords: nonlinear filter generator, algebraic attack, direct sum decomposition, characteristic polynomial, success probability.

1 Introduction

The nonlinear filter generator (NLFG) consists of an n-bit linear feedback shift register (LFSR) and a Boolean function $g: GF(2)^n \to GF(2)$, called filter function, whose n inputs are taken from some shift register stages, called taps, to produce the $GF(2)$ keystream sequence $z = z_0, z_1, z_2, \cdots$. It can either be used to produce the key stream itself [4] or as a building block in a more complex stream cipher system [16]. In this paper all sequence elements are considered over the field $GF(2)$ which consists of the two elements $\{0, 1\}$.

It is known that any encryption map between finite dimensional vector spaces over a finite field is polynomial. Thus one may represent the task of breaking a cryptosystem by the problem of solving a multivariate polynomial system of equations over a finite field. Such techniques are usually called algebraic attacks and have been deemed a powerful tool to cryptanalyze many stream ciphers previously believed very secure. All stream ciphers, especially those with linear feedback are concerned about algebraic attacks.

A major parameter which influences the complexity of the algebraic attack is the degree of the underlying equation system. It is known for a long time that those nonlinear functions involved in stream ciphers which implement the encryption map should have high degrees. One interesting line in the cryptography community recently is to find efficient methods of decreasing the degree of the final equation system to be solved[1, 2, 5, 7], especially in the literature about the attacks against the NLFG [1, 2, 5–7, 15].

Prior Work. For an NLFG, the encryption map consists of the linear feedback of the LFSR and the nonlinear filter function. Its degree is equal to the degree of the latter, which is usually high. An equation system constructed directly from that map is hard to solve.

Traditional algebraic attack methods overcome this by constructing low-degree equations from low-degree multiples of the Boolean function [5], whereas fast algebraic attacks reduce unknowns of original equations by combining them linearly [7]. Both attacks treat equations as random and thus cannot calculate the coefficients of final low-degree equations off-line[15]. Rønjom-Helleseth attacks overcome that by analyzing the output sequence of the NLFG [15]. Given the low-degree equations generated via the fast algebraic attack method in [7], Rønjom and Helleseth showed that the coefficients of these low-degree equations can be computed in an off-line manner by using $LC \times n$ output bits of the NLFG, where LC is the linear complexity of the output sequence. With that pre-computation Rønjom-Helleseth attacks become one of the most efficient algebraic attacks on the NLFG.

Our Contributions. This paper concerns how to construct low-degree algebraic equations by using direct sum decomposition of sequences. We show that the output sequence of an NLFG can be decomposed into a direct sum of some linear recurring sequences. If we call the output stream of the LFSR as the LFSR sequence then one of those decomposed sequences is just the LFSR sequence shifted by c positions where the value c is independent of the initial state of the NLFG. This leads to linear equations between the output bits and the initial state of the NLFG. We show that the degree of these equations is one and the complexity of substituting and solving them is of the same order as that of the RH attack.

We design an algebraic attack based on those linear equations. In this algebraic attack, we can pre-compute the coefficients of the linear equation system. Compared with the RH method, our trick has the advantage that the underlying calculation only requires LC off-line bits, where LC is the linear complexity of the output sequence. Concrete comparison with known prior work shows our attack is amongst the most efficient algebraic methods applicable to NLFG. Refer to Table 1 and Table 2 in Section 4 for more details.

Different from most previous work [2, 5–7] that estimates via experiment simulations the probability of successful recovery of the LFSR initial state, we analyze the probability from a theoretical point of view. By employing our attack against any nonlinear filter generator, one can succeed in recovering the LFSR initial state with probability $1 - 2^{-\phi(2^n-1)}$ ($\phi(\cdot)$ is the Euler function), which is significantly greater than the success probability $1-2^{-n}$ of the Rønjom-Helleseth attack [15].

Roadmap. This paper is organized as follows. Section 2 gives some necessary definitions and notations for the paper. In Section 3 we use direct sum decomposition to distinguish the shifted sequence of the LFSR in the NLFG. Section 4 presents an efficient algebraic attack against the NLFG and discusses its success probability and time/space complexity. Section 5 concludes the paper.

2 Preliminaries

In this section we give some necessary definitions and notations about linear recurring sequences and sequences spaces. While this paper considers the binary case, the results can be extended to other finite fields. See [11] for a thorough discussion.

Let $\underline{s} = s_0, s_1, \ldots$ be a binary sequence. Let $\mathbb{F}_2$ denote the binary field. If the sequence $\underline{s}$ satisfies the following relation, for any positive integer t,

$$s_{t+n} = a_{n-1}s_{t+n-1} + a_{n-2}s_{t+n-2} + \ldots + a_0 s_t$$

with $a_0, a_1, \ldots, a_{n-1} \in \mathbb{F}_2$ then $\underline{s}$ is called a linear recurring sequence. The polynomial $f(x) = x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_0 \in \mathbb{F}_2[x]$ is called a characteristic polynomial of $\underline{s}$. The companion matrix of the polynomial is defined as

$$A = \begin{pmatrix} 0\,0\,0\ldots 0 & a_0 \\ 1\,0\,0\ldots 0 & a_1 \\ 0\,1\,0\ldots 0 & a_2 \\ \vdots\,\vdots\,\vdots \quad \vdots & \vdots \\ 0\,0\,0\ldots 1 & a_{n-1} \end{pmatrix} \in \mathbb{F}_2^{n\times n}.$$

For any state vector $\boldsymbol{s_t} \triangleq (s_t, s_{t+1}, \ldots, s_{t+n-1})$, we have $\boldsymbol{s_t} = \boldsymbol{s_0}A^t$.

Let $\mathrm{Tr}_1^n(x) : \mathbb{F}_{2^n} \to \mathbb{F}_2$ be the trace function. If the characteristic polynomial $f(x)$ is irreducible and α is one of its root over the extensive field $\mathbb{F}_2^n$, then we have

$$s_t = \mathrm{Tr}_1^n(\beta\alpha^t), \text{for some } \beta \in \mathbb{F}_2^n .$$

It is also known that if $f(x)$ is primitive, then $\underline{s}$ is an m-sequence.

Given a reducible polynomial $f(x) = \prod_{i=1}^{d} f_i(x)$, where $f_i(x), i = 1, ..., d$ are co-prime to each other, the sequence $\underline{s}$ with characteristic polynomial $f(x)$ can be viewed as a direct sum of the sequences whose characteristic polynomials are $f_i(x)$ [11]. $\forall b \in \mathbb{F}_2$ and the sequences $\underline{v} = v_0, v_1, \cdots$ and $\underline{w} = w_0, w_1, \cdots$, define

$$\underline{v} + \underline{w} = v_0 + w_0, v_1 + w_1, \cdots$$

and

$$b \cdot \underline{v} = bv_0, bv_1, \cdots$$

then the sequences with same characteristic polynomial form a vector space over $\mathbb{F}_2$ with zero element $\underline{0} = 0, 0, \cdots$.

Let $S(f(x))$ be the vector space of sequences with characteristic polynomial $f(x)$. Then $S(f(x))$ is a direct sum of $S(f_1(x)), S(f_2(x)), \dots, S(f_d(x))$ where $f_1(x), f_2(x), \dots, f_d(x)$ are $d \geq 2$ co-prime polynomials such that $f(x) = f_1(x)f_2(x)\cdots f_d(x)$. Thus the sequence $\underline{s}$ with characteristic polynomial $f(x)$ is a direct sum of sequences $\underline{u_i} \in S(f_i(x)), i = 1, 2, \dots, d$.

3 Using Direct Sum Decomposition to Distinguish the Shifted Sequence

In this section we show the output sequence of an NLFG can be expressed as the direct sum of some linear recurring sequences one of which is very special in the sense that it is the shifted LFSR sequence of the NLFG. More interestingly, we show that the shift value is not affected by the initial state of the NLFG.

Let us consider an NLFG that consists of an LFSR of length n and a filter function g of degree d and produces a $GF(2)$ keystream sequence $\underline{z} = z_0, z_1, z_2, \cdots$. Denote for any integer ℓ by $wt(\ell)$ its binary Hamming weight of ℓ. Next we will factorize the characteristic polynomial of $\underline{z}$ appropriately so that the special sequence we want via the direct sum decomposition of $\underline{z}$ would be obtained. More specifically, let $f_1(x)$ be the primitive feedback polynomial of the LFSR and α be one of the roots of $f_1(x)$ and set $f_i(x), 2 \leq i \leq d$, as the following:

$$f_i(x) \triangleq \prod_{wt(\ell)=i} (x + \alpha^\ell).$$

The polynomials $f_i(x), i = 2, 3, \cdots, d$ are co-prime factors of the characteristic polynomial of $\underline{z}$ and the sequence $\underline{z}$ can be decomposed as below.

Lemma 1. *The output sequence $\underline{z}$ is a direct sum of sequences $\underline{u_i} \in S(f_i(x)), i = 1, 2, \dots, d$.*

Proof. The element α is primitive over $\mathbb{F}_2^n$. Then $f_i(x) = \prod_{wt(\ell)=i}(x + \alpha^\ell), 2 \leq i \leq d$ are relatively prime. Let $f(x)$ be the product $f_1(x)f_2(x)\cdots f_d(x)$. The polynomial $f(x)$ is a characteristic polynomial of the sequence $\underline{z}$ (See Theorem 1 below). Then $\underline{z} \in S(f(x))$. Therefore the sequence $\underline{z}$ can be expressed as a direct sum of sequences $\underline{u_i} \in S(f_i(x)), i = 1, 2, \dots, d$.

Let us now assume that $\underline{u_1} \neq \underline{0}$. We then consider the sequence vector space $S(f_1(x))$. The polynomial $f_1(x)$ is a primitive polynomial and thus the vector space contains exactly those m-sequence with the same characteristic polynomial $f_1(x)$. Then $\underline{u_1} = u_{1,0}, u_{1,1}, \cdots \in S(f_1(x))$ is an m-sequence. If we denote the state of the LFSR at any time instant t by $(u_t, u_{t+1}, \dots, u_{t+n-1})$ then the sequence $\underline{u} = u_0, u_1, \cdots$ is also an m-sequence in $S(f_1(x))$. Therefore, there exists an integer $c, 0 \leq c \leq 2^n - 2$ such that

$$u_{1,t} = u_{t+c}, \quad t = 0, 1, \cdots . \tag{1}$$

Suppose that e is an integer, $0 \leq e \leq n-1$, we denote by $i_0, i_1, \cdots, i_{e-1}$ some e integers in $R_n = \{0, 1, \ldots, n-1\}$ and by $j_0, j_1, \cdots, j_{e-1}$ some e different integers in R_n. Then for the integers $i_0, i_1, \cdots, i_{e-1}$ and the integers $j_0, j_1, \cdots, j_{e-1}$, set $I = 2^{i_0} + 2^{i_1} + \ldots + 2^{i_{e-1}}$, and

$$\mu_{j_0, j_1, \ldots, j_{e-1}} = \sum_{I=1} \alpha^{2^{i_0} j_0} \alpha^{2^{i_1} j_1} \cdots \alpha^{2^{i_{e-1}} j_{e-1}}.$$

The following theorem presents the significant property of the shift value c in the equation (1).

Theorem 1. *Let the filter function g be of the form:*

$$\begin{aligned} g(x_0, x_1, \ldots, x_{n-1}) = & \ b + b_0 x_0 + \cdots + b_{n-1} x_{n-1} \\ & + b_{01} x_0 x_1 + \cdots + b_{n-2,n-1} x_{n-2} x_{n-1} \\ & \cdots \\ & + b_{01\ldots n-1} x_0 x_1 \cdots x_{n-1} \\ & \textit{for some } b, b_0, \ldots, b_{01\ldots n-1} \in \mathbb{F}_2 \,. \end{aligned} \tag{2}$$

Then we have

$$\alpha^c = \begin{cases} \mu & \text{if } b = 0, \\ \frac{\alpha^{2^n - 1}}{\mu} & \text{if } b \neq 0, \end{cases}$$

where

$$\mu = b_0 \mu_0 + \cdots + b_{n-1} \mu_{n-1} + b_{01} \mu_{0,1} + \cdots + b_{n-2,n-1} \mu_{n-2,n-1} \cdots + b_{01\ldots n-1} \mu_{0,1,\cdots,n-1}.$$

Proof. It suffices to prove this for the case that $g(x_0, x_1, \ldots, x_{n-1}) = x_{j_0} x_{j_1} \cdots x_{j_{e-1}}$ Now the output bit is $z_t = g(u_t, u_{t+1}, \ldots, u_{t+n-1}) = u_{t+j_0} u_{t+j_1} \cdots u_{t+j_{e-1}}$, where $u_t = \mathrm{Tr}_1^n(\beta \alpha^t)$ for some $\beta \in \mathbb{F}_2^n$. Then we have

$$\begin{aligned} z_t &= \mathrm{Tr}_1^n(\beta \alpha^{t+j_0}) \, \mathrm{Tr}_1^n(\beta \alpha^{t+j_1}) \cdots \mathrm{Tr}_1^n(\beta \alpha^{t+j_{e-1}}) \\ &= \sum_{k=0}^{n-1} (\beta \alpha^{t+j_0})^{2^k} \cdot \sum_{k=0}^{n-1} (\beta \alpha^{t+j_1})^{2^k} \cdots \sum_{k=0}^{n-1} (\beta \alpha^{t+j_{e-1}})^{2^k} \\ &= \sum_{i=1}^{d} \sum_{wt(I)=i} (\beta^I \alpha^{It} \cdot (\alpha^{2^{i_0} j_0} \alpha^{2^{i_1} j_1} \cdots \alpha^{2^{i_{e-1}} j_{e-1}}))). \end{aligned} \tag{3}$$

Define $u_{i,t} = \sum_{wt(I)=i} (\beta^I \alpha^{It} \cdot (\alpha^{2^{i_0} j_0} \alpha^{2^{i_1} j_1} \cdots \alpha^{2^{i_{e-1}} j_{e-1}}))$. One can see that the sequence formed by $u_{i,t}, t = 0, 1, \cdots$ is with the characteristic polynomial $f_i(x)$. Therefore those $u_{i,t}, t = 0, 1, \cdots$ are exactly the terms of the sequence $\underline{u_i}$ defined in Lemma 1.

Now consider the sequence $\underline{u_1}$. For $m = 0, 1, \cdots, n-1$, set

$$\lambda_m = \sum_{I=2^m} \alpha^{2^{i_0} j_0} \alpha^{2^{i_1} j_1} \cdots \alpha^{2^{i_{e-1}} j_{e-1}}. \tag{4}$$

Especially, we have $\lambda_1 = \mu_{j_0,j_1,\ldots,j_{e-1}}$. An immediate consequence of (4) is that $\lambda_{m+1} = (\lambda_m)^2$. Then following the definition of $u_{i,t}$, the term of $\underline{u_1}$ at time t is

$$\begin{aligned} u_{1,t} &= \sum_{m=0}^{n-1} \beta^{2^m} \alpha^{t2^m} \lambda_m \\ &= \sum_{m=0}^{n-1} \beta^{2^m} \alpha^{t2^m} (\lambda_1)^{2^m} \\ &= \mathrm{Tr}_1^n(\beta\alpha^t \mu_{j_0,j_1,\ldots,j_{e-1}}). \end{aligned} \tag{5}$$

Comparing this with $u_t = \mathrm{Tr}_1^n(\beta\alpha^t)$, one can conclude that the integer c in the equation (1) satisfies $\alpha^c = \mu_{j_0,j_1,\ldots,j_{e-1}} = \mu$. This ends the proof.

Theorem 1 implies that the sequence $\underline{u_1}$ is a shifted one of $\underline{u}$ where the shift value c relies only on α (or, the feedback function of the LFSR) and the filter function g if $\underline{u_1} \neq \underline{0}$. For simplicity, we view hereafter the zero sequence as a kind of shifted sequence; namely the zero sequence is the shifted sequence for any one by infinite positions.

Corollary 1. *Let Γ_i be the set of (cyclotomic) coset leaders of binary Hamming weight i modulo $2^n - 1$. If a is a positive integer we denote by $\underline{u^a}$ the decimated sequence $u_0, u_a, u_{2a}, \cdots$. Then for $2 \leq i \leq d$, the sequence $\underline{u_i}$ is a sum of the shifted sequences of $\underline{u^a}, a \in \Gamma_i$ and that the shift values are also determined only by α and the filter function g.*

4 Attack Based on Direct Sum Decomposition Against NLFG

Now we describe how to use Theorem 1 to design an algebraic attack against the NLFG. Suppose that the attacker somehow knows the shift value c. Then with the linear recurrence relation of both the sequences $\underline{u}$ and $\underline{u_1}$, he knows how to convert the $(t + c)$th state of $\underline{u_1}$ to the initial state of $\underline{u}$ which is exactly the initial state of the NLFG. This is illustrated in matrix operations as follows.

4.1 The Main Idea

We represent the involved states by vectors. Let $\boldsymbol{u_0}$ be the initial state vector $(u_0, u_1, \ldots, u_{n-1})$ and $\boldsymbol{u_{1,t}}$, the state vector $(u_{1,t}, u_{1,t+1}, \ldots, u_{1,t+n-1})$ of the sequence $\underline{u_1}$, and A the companion matrix of the polynomial $f_1(x)$. Therefore, we have

$$\boldsymbol{u_{1,t}} = \boldsymbol{u_0} A^{t+c}. \tag{6}$$

Assume that $f_0(x)$ of degree $deg(f_0) = LC$ is the minimal polynomial of $\underline{z}$, and $h(x) = h_0x^0 + h_1x^1 + \cdots h_{LC-n}x^{LC-n}$ is the polynomial $f_0(x)/f_1(x)$. Let $\boldsymbol{h}$ be the vector $(h_0, h_1, \ldots, h_{LC-n})$ and Z be the matrix $(\boldsymbol{z_t}, \boldsymbol{z_{t+1}}, \ldots, \boldsymbol{z_{t+LC-n}})^T$

with $\boldsymbol{z_t} = (z_t, z_{t+1}, \ldots, z_{t+n-1})$. By the direct sum decomposition algorithm in [9], we know that

$$\boldsymbol{u_{1,t}} = \boldsymbol{h}Z. \tag{7}$$

Combining the equation (6) with the equation (7), we obtain a linear equation system between the output bits and the initial state

$$\boldsymbol{u_0}A^{t+c} = \boldsymbol{h}Z, \tag{8}$$

with coefficient matrix A^{t+c} and the variables $u_0, u_1, \ldots, u_{n-1}$. An immediate consequence of (8) is that the initial state vector can be computed by

$$\boldsymbol{u_0} = \boldsymbol{h}ZA^{-(t+c)}. \tag{9}$$

Namely, if we can compute the matrix A^{t+c} and the vector $\boldsymbol{h}Z$, then we can recover the initial state $\boldsymbol{u_0}$. The coming sections show the details.

4.2 Computing the Coefficient Matrix A^{t+c}

To compute the coefficient matrix, one can calculate the element μ through its explicit expression in Theorem 1, deduce the value c from μ and do the matrix multiplications. But through the linear equation system (8), we can see that the only use of the shift value c is to compute the coefficient matrix. In other words, the deduction of the value c is unnecessary if we have an alternative to do the computation. In fact such an alternative does exist by diagonalizing the companion matrix A. This alternative can also be used to determine the element μ. Here we use this alternative to compute the coefficient matrix A^{t+c}.

Let D be the diagonal matrix $\mathrm{diag}(\alpha, \alpha^2, \ldots, \alpha^{2^{n-1}}) \in \mathbb{F}_{2^n}^{n\times n}$. The elements $\alpha, \alpha^2, \ldots, \alpha^{2^{n-1}}$ are roots of the polynomial $f_1(x)$ and thus are the eigenvalues of A. By diagonalizing the matrix A, one can find a nonsingular matrix $P \in \mathbb{F}_{2^n}^{n\times n}$ such that

$$A = PDP^{-1}. \tag{10}$$

Combining the equation (7) and the equation (10), we have

$$\boldsymbol{u_0}A^tP \cdot D^c = \boldsymbol{h}ZP. \tag{11}$$

Suppose γ_1 and γ_2 are the first elements of the row vectors $\boldsymbol{u_0}A^tP$, $\boldsymbol{h}ZP$ respectively. Then we have $\mu = \gamma_1/\gamma_2$.

As shown in Theorem 1, the element μ is a constant independent of the initial state vector $\boldsymbol{u_0}$, which implies that μ can be computed from any initial state $\boldsymbol{u_0^{(1)}}$ and its corresponding output $Z^{(1)}$. Therefore, to compute μ, we can just feed into the NLFG some selected initial state, obtain the corresponding output bits and do the subsequent calculations.

Despite that the element μ has been known already, we can not calculate the coefficient matrix A^{t+c} from the $(t+c)$th power of A because it is infeasible to deduce c from $\mu = \alpha^c$ due to the hardness of discrete logarithm problem. But the diagonalization of A enables its $(t+c)$ times multiplication from the $(t+c)$

times multiplication, or the c times multiplication of its eigenvalue α which is just the element μ. More precisely, from the diagonalization equation (10), we have

$$A^{t+c} = A^t \cdot PD^cP^{-1} = A^t \cdot P\,\mathrm{diag}(\mu, \mu^2, \ldots, \mu^{2^{n-1}})P^{-1}. \tag{12}$$

Moreover, we can just replace for the element μ its inverse if we want to compute directly the matrix $A^{-(t+c)}$ for the initial state recovery equation (9).

The following algorithm 1 presents the concrete steps of computing A^{t+c}. Note that all the steps can be performed in an off-line manner (i.e., without the knowledge of the output bits to be attacked), and that the coefficient matrix needs only to be computed once for any output sequence of the same NLFG. Thus Algorithm 1 can be seen as the precomputation phase of the algebraic attack described in the next section.

```
Algorithm 1: Pre-calculation of Coefficient Matrix
Inputs: the time instant $t$; the companion matrix $A$ and the roots $\alpha, \alpha^2, \ldots, \alpha^{2^{n-1}}$ of the polynomial $f_1(x)$.
Outputs: the coefficient matrix $A^{t+c}$.

1. Calculate the minimal polynomial $f_0(x)$ and $h(x) = f_0(x)/f_1(x)$. Suppose $h(x) = h_0x^0 + h_1x^1 + \cdots h_{LC-n}x^{LC-n}$, set $\boldsymbol{h} = (h_0, h_1, \ldots, h_{LC-n})$.
2. Select a initial state $u_0^{(1)}, u_1^{(1)}, \ldots, u_{n-1}^{(1)}$ and generate $LC$ bits $z_0, z_1, \ldots, z_{LC-1}$ by the NLFG. Substitute them into the vector $\boldsymbol{u}_\mathbf{0}^{(\mathbf{1})}$ and the matrix $Z^{(1)}$ respectively.
3. Calculate the first element $\gamma_1^{(1)}$ of the vector $\boldsymbol{u}_\mathbf{0}A^tP$ and the first element $\gamma_2^{(1)}$ of $\boldsymbol{h}ZP$.
4. Compute $\mu = \gamma_1^{(1)}/\gamma_2^{(1)}$.
5. Diagonalize $A$ to find a nonsingular $P$ such that $A = P\,\mathrm{diag}(\alpha, \alpha^2, \ldots, \alpha^{2^{n-1}})P^{-1}$.
6. Calculate the coefficient matrix $A^{t+c} = P\,\mathrm{diag}(\mu\alpha^t, \mu^2\alpha^{2t}, \ldots, \mu^{2^{n-1}}\alpha^{2^{n-1}t})P^{-1}$ and output it.
```

In Algorithm 1, calculating the minimal polynomial $f_0(x)$ is the most dominant operation and takes the complexity of $O(LC(n(\log n)^2 + (\log(LC))^3))$, according to [10]. Therefore the time complexity of Algorithm 1 is $O(LC(n(\log n)^2 + (\log(LC))^3))$. For the data complexity, we need only LC off-line bits to fulfill the algorithm.

4.3 Attack Against the NLFG

After obtaining the coefficient matrix A^{t+c}, we are close to the attack against the NLFG. In fact, what's remaining is to solve a linear equation system. We summarize the whole algebraic attack in Algorithm 2.

We assume that $hZ = \boldsymbol{u}_{\mathbf{1,0}} \neq \mathbf{0}$ in previous sections to launch the attack. If it is not the case we may pick an integer $a \neq 1$ such that $(a, 2^n - 1) = 1$. Then

Algorithm 2: Algebraic Attack on the NLFG

Inputs: the output sequence $\underline{z} = z_0, z_1, \ldots$; the feedback polynomial of the LFSR $f_1(x)$; the Boolean function $g(x_0, x_1, \ldots, x_{n-1})$.
Outputs: the initial state $u_0, u_1, \ldots, u_{n-1}$.

```
1. Pre-compute the coefficient matrix by Alg. 1 and store the vector $\boldsymbol{h}$.
2. Substitute consecutive $LC$ bits of $\underline{z}$ and calculate $\boldsymbol{h}Z$.
3. Solve the equations $\boldsymbol{u_0}A^{t+c} = \boldsymbol{h}Z$ and output the result.
```

the initial state of $\underline{u^a}$ can be converted into the initial state of $\underline{u}$ by the discrete fourier transform [8]. By Corollary 1 the shifted sequence of $\underline{u^a}$ falls into the direct sum decomposition of $\underline{z}$ and the shift value can be pre-determined in a similar way. Therefore we can still recover the initial state $\boldsymbol{u_0}$ even if $\underline{u_1} = \underline{0}$.

4.4 Results

To see how efficient our attack against the NLFG is, we study the success probability of our attack in recovering the initial state and the time and space complexities of the attack.

Success Probability

Now we analyze the success probability of recovering the initial state by using the proposed attack against the NLFG. Suppose that with probability 1/2, that shifted sequence of $\underline{u^a}, (a, 2^n - 1) = 1$ in the direct sum decomposition is not the zero sequence. Then one can succeed in recovering the initial state with a probability at most

$$1 - \frac{1}{2^{\phi(2^n-1)}}, \tag{13}$$

which is sufficiently high for the attacker. Herein, $\phi(\cdot)$ is the Euler function. The coefficient matrix of the underlying algebraic system is the power of a state transition matrix for some LFSR and is thus non-singular. As long as there exists an integer a such that $(a, 2^n - 1) = 1$ and the shifted sequence of $\underline{u^a}$ is not $\underline{0}$, the initial state is exactly the unique solution of the algebraic system.

Therefore, the probability in (13) is the exact value of the success probability of recovering the initial state. We are inclined to address here that most of the existing algebraic attacks only estimate via experiment simulations the probability of successful recovery of the LFSR initial state. Although Rønjom and Helleseth also analyzed the probability of successful attack in [15] from a theoretical point of view, one can employ our method to recover the LFSR initial state with a probability significantly greater than the success probability

$$1 - \frac{1}{2^n} \tag{14}$$

in the Rønjom-Helleseth attack.

Complexity and Comparison

Consider the performance of the Algorithm 2. The space complexity of Algorithm 2 is $O(n^2 + LC)$. The time complexity and the data complexity are $O(LC)$ and $O(LC)$ respectively. Table 1 compares the known algebraic attacks applicable to the NLFGs in the literature. In the table, we denote by D_1 the sum $\sum_{i=0}^{d/2} \binom{n}{i}$ and by D_2 the sum $\sum_{i=2}^{d} \binom{n}{i}$. W.l.o.g., $LC \leq D_2$. For the time complexity comparison, both pre-computation and on-line computation are concerned. Note that the pre-computation stage of the algebraic attack in [5] relies heavily on the complexity of finding low-degree multiples of the filter function of the NLFG and thus is uncertain to be measured [3]. Table 1 shows that our proposed attack is as efficient as the RH algebraic attack and is more efficient than the fast algebraic attack and the improved fast algebraic attack on the NLFG.

Table 1. Comparison Among Known Algebraic Attacks on NLFG

	Data	Space	Time	
			Precomputation	Computation
AA[5]	$O(D_1)$	$O((D_1)^2)$	uncertain	$O((D_1)^{\omega})^2$
FAA[7][10][1]	$O(LC)$	$O(n^2)$	$O((LC)^2)$	$O(n^2 LC)$
Improved FAA for the NLFG[10]	$O(D_2)$	$O(n^2 + LC)$	$O(D_2(n(\log n)^2 + (\log D_2)^3))$	$O(n \log n D_2)$
RH[15]	$O(LC)$	$O(n^2 + LC)$	$O(LC(n(\log n)^2 + (log(LC))^3))$	$O(LC)$
Alg.2	$O(LC)$	$O(n^2 + LC)$	$O(LC(n(\log n)^2 + (log(LC))^3))$	$O(LC)$

[1] Part of the complexity result in [7] is corrected by [10].
[2] $\omega \approx 2.807$.

As the RH attack in [15] is amongst the most efficient methods sofar, we compare it with the method proposed in this paper step by step as shown in Table 2. Both methods use pre-computation to improve the performance of the attacks. But we observe that the output sequence of the NLFG has the very special property that one direct sum decomposition of the sequence includes the shifted LFSR sequence of the NLFG and that shift value is not affected by the initial state of the NLFG. This motivation of our attack result in the performance differences between the RH attack and ours, i.e., far fewer output bits and significantly less matrix computation are required by the proposed method. Moreover, the success probability of the proposed method is much greater than that in the RH attack, as shown in the equations (13) and (14).

Table 2. Comparison of RH Algebraic Attack and Algorithm 2

	Precomputation				Computation	
	Calculate $f_0(x)$	Generate D_0 bits [1]	Calculate $\boldsymbol{h}Z$	Diagonalize and multiply $n \times n$ matrices	Calculate $\boldsymbol{h}Z$	Solve n linear equations
RH[15]	once	n times	n times	-	once	once
Alg.2	once	once	once	C times [2]	once	once

[1] $D_0 = D_2 + n$.
[2] C is a constant.

5 Conclusion

Based on direct sum decomposition, a special property of the NLFG output sequence has been explored and exploited to provide new methods of decreasing the degree of the algebraic equation system for the algebraic attack. We show that the output sequence can be decomposed in a direct sum of a series of sequences, one of which is a shifted one of the LFSR sequence. The shift value is a constant independent of the NLFG initial state and thus can be pre-determined. The resulting equation system is linear and thus easy to be solved. Moreover, the coefficient matrix of the algebraic system, which depends just on the shift value, needs only to be calculated once for the NLFG. The total complexity of our algebraic attack based on that equation system is the same as that of the RH algebraic attack, one of the most efficient algebraic methods applicable to the NLFG sofar. Theoretical analysis shows that our attack requires fewer operations and achieves higher success probability than the RH algebraic attack.

References

1. Armknecht, F., Krause, M.: Algebraic Atacks on Combiners with Memory. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 162 – 175. Springer, Heidelberg (2003)
2. Armknecht, F., Ars, G.: Introducing a New Variant of Fast Algebraic Attacks and Minimizing Their Successive Data Complexity. In: Dawson, E., Vaudenay, S. (eds.) Mycrypt 2005. LNCS, vol. 3715, pp. 16 – 32. Springer, Heidelberg (2005)
3. Canteaut, A.: Open Problems Related To Algebraic Attacks on Stream Ciphers. In: Ytrehus, ø. (ed.) WCC 2005. LNCS, vol. 3969, pp. 120 – 134. Springer, Heidelberg (2006)
4. Courtois, N.: Higher Order Correlation Attacks, XL algorithm and Cryptanalysis of Toyocrypt. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 182 – 199. Springer, Heidelberg (2003)
5. Courtois, N., Meier, W.: Algebraic Attacks on Stream Ciphers with Linear Feedback. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 345-359. Springer, Heidelberg (2003)

6. Courtois, N.: Cryptanalysis of SFINKS. In: ICISC 2005. Cryptology ePrint Archive Report 2005/243 (2005), http://eprint.iacr.org/
7. Courtois, N.: Fast Algebraic Attacks on Stream Ciphers with Linear Feedback. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 176-194. Springer, Heidelberg (2003) Heidelberg (2006)
8. Golomb, S.W., Gong, G.: Signal Design for Good Correlation: For Wireless Communication, Cryptography and Radar. Cambridge University Press, Cambridge (2005)
9. Guang Gong, Ronjom, S., Helleseth, T., Honggang Hu: Fast Discrete Fourier Spectra Attacks on Stream Ciphers. IEEE Trans. Inform. Theory 57(8), 5555-5565 (2011)
10. Hawkes, P., Rose, G.: Rewriting Variables: The Complexity of Fast Algebraic Attacks on Stream Ciphers. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 390-406. Springer, Heidelberg (2004)
11. Lidl R., Niederreiter H.: Finite Fields, Encyclopedia of Mathematics and its Applications, 2nd ed., vol. 20. Cambridge University Press, Cambridge (1997)
12. Meier, W., Pasalic, E., Carlet, C.: Algebraic Attacks and Decomposition of Boolean Functions. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 474-491. Springer, Heidelberg (2004)
13. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996).
14. Rueppel, R.A.: Analysis and Design of Stream Ciphers. Springer, Heidelberg (1986)
15. Rønjom, S., Helleseth, T.: A New Attack on the Filter Generator. IEEE Trans. Inform. Theory 53(5), 1752-1758 (2007)
16. Simpson, L., Dawson, E., Golic, J., Millan, W.: LILI Keystream Generator. In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 248 – 261. Springer, Heidelberg (2001)