On Attacks on Filtering Generators Using Linear Subspace Structures

Sondre Rønjom, Guang Gong* and Tor Helleseth

The Selmer Center,
Department of Informatics, University of Bergen
PB 7803
N-5020 Bergen, Norway
* Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario N2L 3G1, CANADA

Abstract. The filter generator consists of a linear feedback shift register (LFSR) and a Boolean filtering function that combines some bits from the shift register to create a key stream. A new attack on the filter generator has recently been described by Rønjom and Helleseth [6]. This paper gives an alternative and extended attack to reconstruct the initial state of the LFSR using the underlying subspace structure of the filter sequence. This improved attack provides further insight and more flexibility in performing the attack by Rønjom and Helleseth. The main improvements are that it does not use the coefficient sequences that were fundamental in the previous attack and works better in the unlikely cases when the original attack needed some modifications.

Keywords filter generator, m-sequences, stream ciphers.

1 Introduction

The filter generator uses a primitive linear feedback shift register of length n that generates a maximal linear sequence (an m-sequence) $\{s_t\}$ of period $2^n - 1$ satisfying a recursion with a characteristic polynomial $h(x) \in \mathbb{F}_2[x]$ of degree n being a primitive polynomial with primitive zeroes α^{2^i} for $i = 0, 1, \ldots, n-1$.

At each time t, a key stream bit b_t is calculated as a function of certain bits in some positions $(e_0, e_1, \ldots, e_{m-1})$ in the LFSR state $(s_t, s_{t+1}, \ldots, s_{t+n-1})$ at time t using a Boolean function $f(x_0, x_1, \ldots, x_{m-1})$ of degree d in $m \leq n$ variables. The key stream is defined by

$$b_t = f(s_{e_0+t}, s_{e_1+t}, \ldots, s_{e_{m-1}+t}).$$

Since s_t is a linear combination of the bits in the initial state $(s_0, s_1, \ldots, s_{n-1})$ this leads to an equation system

$$b_t = f_t(s_0, s_1, \ldots, s_{n-1}) \text{ for } t = 0, 1, \ldots$$

which has the initial state of the LFSR as a solution.

In a recent paper Rønjom and Helleseth [6] present a new attack that reconstructs the initial state $(s_0, s_1, \ldots, s_{n-1})$ of the binary filter generator using D key stream bits with complexity $O(D)$, where $D = \sum_{i=1}^{d} \binom{n}{i}$, after a precomputation of complexity $O(D(log_2 D)^3)$. If L is the linear complexity of the key stream then sometimes D can be replaced by L in these complexity estimates.

The underlying method in [6] is based on the observation that since

$$s_t = \sum_{i=0}^{n-1} (c_i \alpha^t)^{2^i} = Tr(c_0 \alpha^t)$$

the key stream b_t can be generated by a characteristic polynomial $p(x)$ with zeroes among α^J, where the Hamming weight of the binary representation of J, denoted $wt(J)$, obey $1 \leq wt(J) \leq d = deg(f)$. Therefore, we have

$$b_t = \sum_{\beta} d_\beta \beta^t$$

where $p(\beta) = 0$ and $d_\beta \in \mathbb{F}_{2^n}$. This polynomial can be constructed in the precomputation phase with complexity $O(D(log_2 D)^3)$([3]).

The attack in [6] selects an irreducible polynomial $k(x)$ of degree n (in [6] the LFSR polynomial $h(x)$ was selected) and define the polynomial $p^*(x) = \frac{p(x)}{k(x)}$. The authors apply the shift operator $p^*(x) = \sum_{j=0}^{L-n} p_j x^j$ to the key stream b_t i.e., compute

$$\sum_{j=0}^{L-n} p_j b_{t+j} = \sum_{j=0}^{L-n} p_j f_{t+j}(s_0, s_1, \ldots, s_{n-1})$$

and show that this almost always leads to a nonsingular linear equation system in the unknowns $s_0, s_1, \ldots, s_{n-1}$.

The main reason for this is the simple observation that

$$\sum_{j=0}^{L-n} p_j b_{t+j} = \sum_{\beta} d_\beta p^*(\beta) \beta^t$$

where the summation now is only over the n zeros of $k(x)$ which implies that the right hand side for the case when $k(x) = h(x)$ can be written as $Tr(d_\alpha \alpha^t)$, a linear combination of bits in the initial state. In the very unlikely case that the system is trivial which happens when $d_\alpha = 0$ one needs to do some modifications. In [6] a non-irreducible polynomial with $d_\alpha \neq 0$ was suggested for $k(x)$, while a better choice in this case would be, for example, to use a different irreducible polynomial $k(x)$ with this property.

In this paper we will take advantage of the fact that the filtering sequence can be written in a unique way as a sum of sequences with characteristic polynomials being all irreducible divisors of $p(x)$. Using a suitable shift operator to the key

stream we can find any component sequence and find relations that determine the initial state of the LFSR for any key stream b_t. Since this paper only needs the linear subspace structure of the filter generator, the results can be directly extended to a combination generator as well.

2 Notations and Preliminaries

A. The Left Shift Operator

Let $q = p^h$ for a prime integer p and a positive integer h. We denote a finite field of q elements by $\mathbb{F}_q$. Let $V(\mathbb{F}_q)$ be a set consisting of all infinite sequences whose elements are taken from $\mathbb{F}_q$, i.e.,

$$V(\mathbb{F}_q) = \{\mathbf{s} = (s_0, s_1, \cdots) \,|\, s_t \in \mathbb{F}_q\}.$$

Then $V(\mathbb{F}_q)$ is a linear space over $\mathbb{F}_q$. Let $\mathbf{s} = (s_0, s_1, s_2, \cdots) \in V(\mathbb{F}_q)$ whose elements satisfy the linear recursive relation

$$s_{t+n} = \sum_{i=0}^{n-1} c_i s_{t+i}, t = 0, 1, \cdots$$

Here $\mathbf{s}$ is referred to as a linear recursive sequence or it is a linear feedback shift register sequence generated by $h(x) = x^n - (c_{n-1}x^{n-1} + \cdots + c_0)$, and $h(x)$ is called a characteristic polynomial of $\mathbf{s}$. Furthermore, the characteristic polynomial of $\mathbf{s}$ with the smallest degree is called the minimal polynomial of $\mathbf{s}$.

The (left) shift operator E is defined as follows:

$$E\mathbf{s} = (s_1, s_2, s_3, \cdots) \text{ and } E^t\mathbf{s} = (s_t, s_{t+1}, s_{t+2}, \cdots), t \geq 1.$$

By convention, we write $E^0\mathbf{s} = I\mathbf{s} = \mathbf{s}$, where I is the identity transformation on $V(\mathbb{F}_q)$. Then $h(E)\mathbf{s} = 0$ where $0 = (0, 0, \cdots)$ is the zero sequence in $V(\mathbb{F}_q)$. For any non-zero polynomial $h(x) \in \mathbb{F}_q[x]$, we use $G(h)$ to represent the set consisting of all sequences in $V(\mathbb{F}_q)$ with

$$h(E)\mathbf{s} = 0.$$

Since $h(E)$ is also a linear transformation, $G(h)$ is a subspace of $V(\mathbb{F}_q)$.

B. m-sequences and Trace Representation

Let $h(x)$ be a primitive polynomial of degree n, and let α be a root of $h(x)$ in $\mathbb{F}_{q^n}$. Then the trace representation of $\mathbf{s}$ is given by

$$s_t = Tr(\beta\alpha^t), t = 0, 1, \ldots, \beta \in \mathbb{F}_{q^n}$$

where $Tr(x)$ denotes the trace function $Tr(x) = \sum_{i=0}^{n-1} x^{q^i}$ which is a map from $\mathbb{F}_{q^n}$ to $\mathbb{F}_q$

C. Filter Generators

Select $m \leq n$ integers: $0 \leq e_0 \leq e_1 \leq \ldots \leq e_{m-1} < n$, and a polynomial function $f(x_0, x_1, \ldots, x_{m-1}) \in \lambda = \mathbb{F}_q[x_0, x_1, \ldots, x_{m-1}]/(x_i^2 + x_i)_{0 \leq i < m}$ of degree d that can be written in the form

$$f(x_0, x_1, \cdots, x_{m-1}) = \sum c_{i_0,i_1,\cdots,i_{m-1}} x_{i_0}^{i_0} x_{i_1}^{i_1} \cdots x_{i_{m-1}}^{i_{m-1}},\ c_{i_0,i_1,\cdots,i_{m-1}} \in \mathbb{F}_q.$$

A sequence $\mathbf{b} = \{b_t\}$ whose elements are defined by a function of

$$b_t = f(s_{e_0+t}, s_{e_1+t}, \ldots, s_{e_{m-1}+t}) = f_t(s_0, s_1, \ldots, s_{n-1}), t = 0, 1, \ldots,$$

is called a filtering sequence and the function $f(x_0, x_1, \ldots, x_{m-1})$ is called a filtering function.

D. DFT and Inverse DFT for the binary case

Let $\{a_t\}$ be a sequence over $\mathbb{F}_q$ with period $N = q^n - 1$. Recall that α is a primitive element in $\mathbb{F}_{q^n}$. Then the *(discrete) Fourier Transform (DFT)* of $\{a_t\}$ is defined by

$$A_k = \sum_{t=0}^{N-1} a_t \alpha^{-tk},\ k = 0, 1, \ldots, N-1.$$

The inverse DFT (IDFT) is given by

$$a_t = \frac{1}{N} \sum_{k=0}^{N-1} A_k \alpha^{kt},\quad t = 0, 1, \ldots, N-1.$$

The sequence $\{A_k\}$ is referred to as a *spectral sequence* of $\{a_t\}$. Let $A(x) = \sum_{k=0}^{N-1} A_k x^k$. Then $A(x)$ can be written as

$$A(x) = \sum_k Tr_1^{m_k}(A_k x^k)$$

where the k's are (cyclotomic) coset leaders modulo N, and $m_k \,|\, n$ is the length of the coset which contains k. This is called a *trace representation* of $\{a_t\}$.

For more general treatments on minimal polynomials of the elements in a finite field or a periodic sequence, and the DFT of sequences, the reader is referred to [1].

3 Minimal Polynomials with Constrained Weights

In this section, we show the linear subspace structure of the filtering sequence. For a positive integer i, we define

$$H(i) = \sum_{j=0}^{n-1} i_j, i = \sum_{j=0}^{n-1} i_j q^j, 0 \leq i_j < q$$

which is referred to as the weight of i. Note that $H(i)$ is the usual Hamming weight of i when $q = 2$. It is known that the zeroes of the minimal polynomial of the filtering sequence $\mathbf{b}$ is a subset of the following set:

$$\Omega(d) = \{\alpha^i \mid H(i) \leq d\} \tag{1}$$

where $d = deg(f)$ $(\leq m)$.

Let $g_{\alpha^i}(x)$ be the minimal polynomial of α^i over $\mathbb{F}_q$ which is given by

$$g_{\alpha^i}(x) = \prod_{j=0}^{n_i-1} (x - \alpha^{i \cdot q^j}),$$

where n_i is the size of the coset C_i which is the smallest number satisfying $i \equiv i \cdot q^{n_i} \pmod{q^n - 1}$. Let T be the set consisting of all coset leaders modulo $q^n - 1$ and define

$$T(d) = \{i \in T \mid 1 \leq H(i) \leq d\}.$$

Let $p(x)$ be the polynomial

$$p(x) = \prod_{i \in T(d)} g_{\alpha^i}(x)$$

and $p_i(x)$ the polynomial

$$p_i(x) = \frac{p(x)}{g_{\alpha^i}(x)}.$$

Then $G(p)$ can be written as a direct sum of the subspaces of $V(\mathbb{F}_q)$

$$G(p) = G(g_{\alpha^{t_1}}) \oplus \ldots \oplus G(g_{\alpha^{t_s}})$$

where $t_j \in T(d) = \{t_1, \ldots, t_s\}$. Let $\mathbf{a}_k = \{a_{k,t}\}_{t \geq 0} \in G(g_{\alpha^k}), k \in T(d)$. We have

$$b_t = \sum_{k \in T(d)} d_k a_{k,t}, d_k \in \{0, 1\}. \tag{2}$$

4 Extractors

Let L be the linear span of $\{b_t\}$, and $l = L - n$. From (1), L is upper bounded by the cardinality of $\Omega(d)$, which is $D = \sum_{i=1}^{d} \binom{n}{i}$ for $q = 2$, the binary case. We may write

$$b_t = \sum Tr_1^{n_k}(A_k(\beta\alpha^t)^k)\,, t = 0, 1, \ldots$$

and

$$a_{k,t} = Tr_1^{n_k}(A_k(\beta\alpha^t)^k)\,, t = 0, 1, \ldots$$

For k with $gcd(k, q^n - 1) = 1$, let $p_k(x) = x^l + \sum_{i=0}^{l-1} c_i x^i, c_i \in \mathbb{F}_q$. For avoiding the use of a double index, we denote

$$\mathbf{a}_k = \{a_{k,t}\}_{t \geq 0} = \{a_t\}_{t \geq 0}.$$

Then we have

$$a_t = Tr(A_k(\beta\alpha^t)^k), t = 0, 1, \ldots.$$

In the following, we show how to separate or extract $\mathbf{a}_k$ from $\mathbf{b} = \{b_t\}$, the filtering sequence. From (2), it follows that

$$\mathbf{b} = \sum_{j=1}^{s} \mathbf{a}_j$$

and thus we have that

$$p_k(E)\mathbf{b} = \sum_{j=1}^{s} p_k(E)\mathbf{a}_j. \tag{3}$$

Note that $p_k(E)\mathbf{a}_j = 0$ if $j \neq k$. Thus (3) above becomes

$$p_k(E)\mathbf{b} = p_k(E)\mathbf{a}_k. \tag{4}$$

If we let

$$u_t = a_{l+t} + \sum_{i=0}^{l-1} c_i a_{i+t}, t = 0, 1, \ldots,$$

we see that

$$p_k(E)\mathbf{a}_k = (u_0, u_1, \ldots).$$

Going through the details, we have that

$$\begin{aligned} u_t &= a_{l+t} + \sum_{i=0}^{l-1} c_i a_{i+t} \\ &= Tr(A_k(\beta\alpha^{l+t})^k) + \sum_{i=0}^{l-1} c_i Tr(A_k(\beta\alpha^{i+t})^k) \\ &= Tr(A_k\beta^k(a^{lk} + \sum_{i=0}^{l-1} c_i\alpha^{ik})\alpha^{tk}) \\ &= Tr(A_k\beta^k p_k(\alpha^k)\alpha^{tk}) \\ &= Tr(r\alpha^{tk}) \end{aligned}$$

where $r = A_k\beta^k p_k(\alpha^k)$ and $p_k(\alpha^k) \neq 0$ since α^k is not a root of $p_k(x)$. In general, we have therefore shown that

$$u_t = Tr(r\alpha^{tk}), t = 0, 1, \ldots, \tag{5}$$

where $r = A_k\beta^k p_k(\alpha^k)$.

Recall that $g(x)$ is the minimal polynomial of $\mathbf{b} = \{b_t\}$ and $gcd(k, 2^n - 1) = 1$. It follows that

$$\begin{aligned} g_{\alpha^k}(x)|g(x) &\Leftrightarrow (u_0, \ldots, u_{n-1}) \neq 0 \\ &\Leftrightarrow A_k \neq 0. \end{aligned}$$

We call $(u_0, u_1, \cdots, u_{n-1})$ an *extractor* of $\mathbf{b}$.

5 Extract β

Let $(b_0, b_1, \ldots, b_{D-1})$ be known. The goal is to obtain β. This yields the initial state of the LFSR which produces $\mathbf{b}$. From (4) we have that

$$p_k(E)\mathbf{b} = p_k(E)\mathbf{a}_k = (u_0, u_1, \ldots)$$

and so

$$\begin{aligned} u_0 &= \sum_{i=0}^{l} c_i b_i \\ u_1 &= \sum_{i=0}^{l} c_i b_{i+1} \\ &\vdots \\ u_{n-1} &= \sum_{i=0}^{l} c_i b_{i+n-1}. \end{aligned}$$

Thus $(u_0, u_1, \ldots, u_{n-1})$ can be computed from $(b_0, b_1, \ldots, b_{L-1})$. From (5) and $(u_0, \ldots, u_{n-1})$ a system of equations with unknown β is formed

$$\begin{aligned} u_0 = \quad Tr(r) = \quad & r + r^2 + \ldots + r^{2^{n-1}} \\ u_1 = \quad Tr(r\gamma) = \quad & \gamma r + \gamma^2 r^2 + \ldots \gamma^{2^{n-1}} r^{2^{n-1}} \\ \vdots & \\ u_{n-1} = Tr(r\gamma^{n-1}) = \gamma^{n-1} r + \gamma^{(n-1)2} r^2 + \ldots + & \gamma^{(n-1)2^{n-1}} r^{2^{n-1}} \end{aligned}$$

where $\gamma = \alpha^k$.

Let $x_i = r^{2^i}$ and $\alpha_i = \alpha^{k2^i}$ for $i = 0, 1, \ldots, n-1$ and form a matrix M of the form

$$M = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ \alpha_0 & \alpha_1 & \ldots & \alpha_{n-1} \\ \alpha_0^2 & \alpha_1^2 & \ldots & \alpha_{n-1}^2 \\ \vdots & \vdots & \ldots & \vdots \\ \alpha_0^{n-1} & \alpha_1^{n-1} & \ldots & \alpha_{n-1}^{n-1} \end{bmatrix}.$$

Then we have

$$M \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n-1} \end{bmatrix}.$$

Since M is a Vandermonde matrix and $(u_0, u_1, \ldots, u_{n-1})$ is known, by solving this equation system we obtain $x_0 = r$. This gives

$$r = A_k \beta^k p_k(\alpha^k)$$

and therefore

$$\beta^k = r A_k^{-1} [p_k(\alpha^k)]^{-1}$$

where r and $p_k(\alpha^k)$ are known. The remaining task is how to find A_k. Note that $\{A_k\}$ is related to a discrete Fourier transform of $\{b_t\}$, which can be computed through expansion of b_t.

6 How to Compute $\{A_k\}$

In this section, we restrict ourselves to the binary case. For the q-ary ($q > 2$) case, there is a similar result which is omitted here for simplicity. In the binary case, $f(x_0, x_1, \ldots, x_{m-1}) = \sum_{(i_1,\cdots,i_e)} c_{i_1,\ldots,i_e} x_{i_1} \cdots x_{i_e}$. Then

$$b_t = f(s_{e_0+t}, \ldots, s_{e_{m-1}+t}) = \sum_{\underline{i}} c_{i_1,\ldots,i_e} s_{i_1+t} \cdots s_{i_e+t}$$

where $\underline{i} = \{i_1, \ldots, i_e\} \subset \{e_0, \ldots, e_{m-1}\}$. Let

$$y_t = s_{i_1+t} s_{i_2+t} \cdots s_{i_e+t}, t = 0, 1, \ldots$$

be a typical term. Since $s_j = Tr(\beta\alpha^j)$, we expand y_t

$$\begin{aligned} y_t &= Tr(\beta\alpha^{i_1+t}) Tr(\beta\alpha^{i_2+t}) \cdots Tr(\beta\alpha^{i_e+t}) \\ &= [\sum_{j_1=0}^{n-1} (\beta\alpha^{i_1+t})^{2^{j_1}}] \cdots [\sum_{j_e=0}^{n-1} (\beta\alpha^{i_e+t})^{2^{j_e}}] \\ &= \prod_{j=1}^{e} (\sum_{v=0}^{n-1} \alpha^{i_j 2^v} x^{2^v}) \\ &= \sum_{v_1, v_2, \ldots, v_e} \alpha^{i_1 2^{v_1} + i_2 2^{v_2} + \cdots i_e 2^{v_e}} x^{2^{v_1} + 2^{v_2} + \cdots 2^{v_e}} \end{aligned}$$

where $x = \beta\alpha^t$.

The following theorem is useful for simplifying the calculations of y_t.

Theorem 1.

$$y_t = \sum_{k \in T(e)} Y_{\underline{i},k} x^k, \underline{i} = (i_1, \ldots, i_e)$$

where

$$Y_{\underline{i},k} = \sum_u \sum_J \alpha^{J(u)}, \; J(u) = \sum_{j=1}^{v} 2^{t_j} h_j \tag{6}$$

where $u = (u_0, u_1, \ldots, u_{n-1})$ *is a solution of*

$$\begin{cases} \sum_{i=0}^{n-1} u_i 2^i \equiv k \pmod{2^n - 1} \\ \sum_{i=0}^{n-1} u_i = e, u_i \geq 0, \end{cases}$$

and J *is a partition of* $\{i_1, \ldots, i_e\}$ *into* v *parts for which the* j*th part has size* u_{t_j} *with*

$$\{t_1, \ldots, t_v\} = \{i \,|\, u_i \neq 0\}, v \leq e$$

and h_j *is the sum of elements in the* j*th part.*

Remark. The inner sum of (6) can be described by the sum of the different permutation terms in the determinant of the following $e \times e$ matrix:

$$E_u = \begin{bmatrix} \overbrace{\alpha^{i_1 2^{t_1}} \cdots \alpha^{i_1 2^{t_1}}}^{u_{t_1}} & \overbrace{\alpha^{i_1 2^{t_2}} \cdots \alpha^{i_1 2^{t_2}}}^{u_{t_2}} & \cdots & \overbrace{\alpha^{i_1 2^{t_v}} \cdots \alpha^{i_1 2^{t_v}}}^{u_{t_v}} \\ \vdots & & & \\ \alpha^{i_e 2^{t_1}} \cdots \alpha^{i_e 12^{t_1}} & \alpha^{i_e 2^{t_2}} \cdots \alpha^{i_e 2^{t_2}} & \cdots & \alpha^{i_e 2^{t_v}} \cdots \alpha^{i_e 2^{t_v}} \end{bmatrix}.$$

In other words, $\sum_J \alpha^{J(u)}$ is equal to the sum of different permutation terms of E_u. When $u_{t_j} = 1$ for all $j = 1, \cdots, v$ ($v = e$ in this case), we have $\sum_J \alpha^{J(u)} = \det E_u$, the determinant of E_u.

Theorem 2. *For*

$$b_t = \sum_{k \in T(d)} Tr_1^{n_k}(A_k(\beta\alpha^t)^k),$$

then

$$A_k = \sum_{\underline{i}: c_{i_1, \ldots, i_e} \neq 0} Y_{\underline{i},k}$$

where $Y_{\underline{i},k}$ *is given by Theorem 1.*

The proofs of these two theorems including the q-ary case, $q > 2$, can be found in [2] or derived from the results in [4] and [5].

Example 1. Consider a filter generator consisting of an m-sequence $\mathbf{s} = \{s_t\}$ generated by $h(x) = x^4 + x + 1 \in \mathbb{F}_2$, where we can write $\{s_t\}$ in terms of the zeroes of $h(x)$ by

$$s_t = Tr(\beta\alpha^t), t = 0, 1, \ldots, \beta \in \mathbb{F}_{2^4},$$

filtered through the simple function

$$f(x_0, x_1, x_3) = x_0 x_1 x_3$$

with $deg(f) = 3$. Let $(e_0, e_1, e_2) = (0, 1, 3)$ be the tapping positions from the register such that the key stream sequence b_t is given by

$$b_t = f_t(s_0, s_1, s_2, s_3) = f(s_{t+e_0}, s_{t+e_1}, s_{t+e_2}) = s_t s_{t+1} s_{t+3}.$$

In this example we assume that

$$b_t = 0001000010\ldots$$

The cosets modulo 15 are

$$\begin{aligned} C_1 &= \{1, 2, 4, 8\} \\ C_3 &= \{3, 6, 12, 9\} \\ C_5 &= \{5, 10\} \\ C_7 &= \{7, 14, 13, 11\}, \end{aligned}$$

so the corresponding polynomials g_{α^i} are

$$\begin{aligned} g_\alpha(x) &= \prod_{i \in C_1} (x + \alpha^i) = x^4 + x + 1 \\ g_{\alpha^3}(x) &= \prod_{i \in C_3} (x + \alpha^i) = x^4 + x^3 + x^2 + x + 1 \\ g_{\alpha^5}(x) &= \prod_{i \in C_5} (x + \alpha^i) = x^2 + x + 1 \\ g_{\alpha^7}(x) &= \prod_{i \in C_7} (x + \alpha^i) = x^4 + x^3 + 1. \end{aligned}$$

Thus in this case $p(x)$ is simply the polynomial

$$p(x) = \prod_{i \in T(3)} g_{\alpha^i}(x) = \sum_{i=0}^{14} x^i = \frac{x^{15} + 1}{x + 1}.$$

We can write b_t in terms of the zeroes of $g(x)$ as

$$b_t = \sum_{k \in T(3)} Tr_1^{n_k}(A_k x^k).$$

Using Theorem 1, we now compute $\{A_k\}$. For $k = 1$, we determine $u = (u_0, u_1, u_2, u_3)$ satisfying $\sum_{i=0}^{3} u_i 2^i \equiv 1 \pmod{15}$ and $\sum_{i=0}^{3} u_i = 3$. There is only one such u, which is $u = (0, 0, 2, 1)$, so we compute

$$\begin{aligned} A_1 = J(u) &= \alpha^{4e_0+4e_1+8e_2} + \alpha^{4e_0+8e_1+4e_2} + \alpha^{8e_0+4e_1+4e_2} \\ &= \alpha^{13} + \alpha^5 + \alpha \\ &= \alpha^{14}. \end{aligned}$$

For $k = 3$, we have two possible values $u \in \{(0, 1, 0, 2), (3, 0, 0, 0)\}$ such that $\sum_{i=0}^{3} u_i 2^i \equiv 3 \pmod{15}$ and $\sum_{i=0}^{3} u_i = 3$. Both values of u lead to values of $J(u)$ being α^4, so

$$A_3 = \alpha^4 + \alpha^4 = 0.$$

For $k = 5$, we have two possible values of $u \in \{(1, 2, 0, 0), (0, 0, 1, 2)\}$, leading to values of $J(u)$ being α^3 and α^{12} respectively, and therefore

$$A_5 = \alpha^3 + \alpha^{12} = \alpha^{10}.$$

For $k = 7$, we find only one value of $u = (1, 1, 1, 0)$, leading to $J(u)$ being α^8, and thus

$$A_7 = \alpha^8.$$

Thus we can now write the sequence b_t as

$$b_t = Tr_1^4(\alpha^{14}x + \alpha^8 x^7) + Tr_1^2(\alpha^{10}x^5)$$

where $x = \beta\alpha^t, t = 0, 1, \ldots$. We have that the linear span of $\mathbf{b}$ is $L = 10$, which is equal to the degree of the polynomial $r(x) = g_\alpha(x)g_{\alpha^5}(x)g_{\alpha^7}(x)$. We may choose $k = 7$ and compute

$$\begin{aligned} p_7(x) &= \frac{r(x)}{g_{\alpha^7}(x)} \\ &= x^6 + x^5 + x^4 + x^3 + 1. \end{aligned}$$

Then we have that $u_i = Tr(r\alpha^{7i})$ where

$$\begin{aligned} r &= A_7\beta^7 p_7(\alpha^7) \\ &= \beta^7\alpha^8\alpha^{14} \\ &= \beta^7\alpha^7. \end{aligned}$$

Then from $b_t = 0001000010\ldots$, we compute $p_7(E)\mathbf{b}$ and obtain

$$u_t = b_{t+6} + b_{t+5} + b_{t+4} + b_{t+3} + b_t$$

for $0 \le t \le 3$ and get $(u_0, u_1, u_2, u_3) = (1, 0, 1, 0)$. Then we compute the matrix M defined by

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 \\ \alpha^7 & \alpha^{14} & \alpha^{13} & \alpha^{11} \\ \alpha^{14} & \alpha^{13} & \alpha^{11} & \alpha^7 \\ \alpha^6 & \alpha^{12} & \alpha^9 & \alpha^3 \end{bmatrix}$$

and M_1 obtained by interchanging column 1 of M with u^T, and equals

$$M_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & \alpha^{14} & \alpha^{13} & \alpha^{11} \\ 1 & \alpha^{13} & \alpha^{11} & \alpha^7 \\ 0 & \alpha^{12} & \alpha^9 & \alpha^3 \end{bmatrix}.$$

Then using M_1, M and u we compute $x_0 = r$ by

$$x_0 = \frac{\det M_1}{\det M} = \alpha^6,$$

and since we also have $x_0 = r = \beta^7\alpha^7$, we are left with solving for β and get $\beta = \alpha^3$. So the initial state is found to be

$$\begin{aligned}(s_0, s_1, s_2, s_3) &= (Tr(\beta), Tr(\beta\alpha), Tr(\beta\alpha^2), Tr(\beta\alpha^3)) \\ &= (Tr(\alpha^3), Tr(\alpha^4), Tr(\alpha^5), Tr(\alpha^6)) \\ &= (1, 0, 0, 1).\end{aligned}$$

7 Which Extractor Can Be Computed Efficiently?

The initial state $(s_0, s_1, \cdots, s_{n-1})$ is given by $s_t = Tr(\beta\alpha^t), t = 0, 1, \cdots, n-1$. From Sections 4-5, we know that we can select any k with $\gcd(k, 2^n - 1) = 1$ and $A_k \neq 0$ to solve for β, to obtain the initial state. The computation of the extractor is to compute A_k and $p_k(\alpha^k)$. For any k with $\gcd(k, 2^n - 1) = 1$, the cost for computing $p_k(x)$ are almost the same. Thus the difference between different k only depends on the computational cost of A_k.

If there, for example, is a term of degree d in the Boolean function and the corresponding tap positions are equally spaced, i.e., leading to a term $y_t = s_{t+i_0}s_{t+i_0+j}\cdots s_{t+i_0+(d-1)j}$ then for $k = 2^d - 1$, $A_k \neq 0$ (see [7]). In this case, let $\alpha_i = \alpha^{2^i j}, i = 0, 1, \cdots, d-1$. From Theorems 1-2, since there is only one solution for u, we have

$$A_{2^d-1} = detE_u = \alpha^{i_0(2^d-1)} det \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_0 & \alpha_1 & \cdots & \alpha_{d-1} \\ \alpha_0^2 & \alpha_1^2 & \cdots & \alpha_{d-1}^2 \\ \vdots & & & \\ \alpha_0^{m-1} & \alpha_1^{d-1} & \cdots & \alpha_{m-1}^{m-1} \end{bmatrix} = \alpha_0^{d-1} \prod_{0 \le i < j < m} (\alpha_j + \alpha_i)$$

the last equality is due to fact that the matrix is a Vandermonde matrix. This is the simplest case for computing A_k. The computation complexity is the cost to compute $\binom{d}{2} = \frac{d(d-1)}{2}$ multiplications of two elements in $\mathbb{F}_{2^n}$, which is about $O(d^2n^2)$ binary multiplications. In general, if the tap positions are not equally spaced, as long as there is one k with $H(k) = d$ such that $A_k \neq 0$. In this case, there is only one solution for u in Theorem 1. Thus $A_k = detE_u$ (see Remark in Section 6). Therefore the computation of A_k for such a k is to compute the determinant of an $d \times d$ matrix E_u whose entries are taken from $\mathbb{F}_{2^n}$. This is the simplest case among all the other k's, since for the other k's where $H(k) < d$, there may be more than one solution for u in Theorem 1.

There is a trade-off between the number of known key stream bits of $\mathbf{b} = \{b_t\}$ and the computation cost in the pre-computation stage for forming an extractor. If we know $D = |\Omega(m)|$ consecutive bits of $\mathbf{b}$, then we do not need to compute the minimal polynomial of $\mathbf{b}$. Thus, we could select k wisely to reduce the cost for this particular A_k in pre-computation. If we only know L consecutive bits where L is the linear span of $\mathbf{b}$, then we have to compute the minimal polynomial of $\mathbf{b}$, which can be done by computing spectra $\{A_k\}$ by the method of Theorem

1 and Theorem 2. In this case, there is no saving for choosing special k, since we need to compute A_k for all $k \in \Gamma(d)$. Note that if we only know L consecutive bits of $\mathbf{b}$, we cannot apply the Berlekamp-Massey algorithm to obtain the minimal polynomial of $\mathbf{b}$ because it requires $2L$ consecutive bits of $\mathbf{b}$. So, the computation cost is counted for computing all the $A_j, j \in \Gamma(d)$. We summarize the linear subspace attack, described in Sections 3-6, with these two different approaches in the pre-computation stage in the following two procedures.

Summary of the Linear Subspace Attack:
Procedure 1: $D = |\Omega(d)|$
Input: $b_0, \cdots, b_{D-1}$
Output: $s_0, \cdots, s_{n-1}$

1. Pre-computation:
   - For k with $\gcd(k, 2^n - 1) = 1$ and $H(k) = d$, compute A_k, and select k such that $A_k \neq 0$.
   - Compute $p_k(x) = \prod_{j \neq k, j \in \Gamma(d)} g_{\alpha^j}(x)$.
2. Compute the first n bits of $p_k(E)\mathbf{b}$, which is $u = (u_0, \cdots, u_{n-1})$.
3. Compute $x_0 = \frac{detM_1}{detM}$ where M is a Vandemonde matrix given by $\alpha_i = \alpha^{k2^i}, i = 0, 1, \cdots, n-1$ (see Section 5) and M_1 is the matrix obtained by replacing the first column of M by u^T.
4. Solve for β from $x_0 = \beta^k A_k p_k(\alpha^k)$.
5. Compute $s_t = Tr(\beta\alpha^t), t = 0, \cdots, n-1$.
6. Return $s_0, \cdots, s_{n-1}$.

Procedure 2: L**, the linear span of** $\mathbf{b}$
Input: $b_0, \cdots, b_{L-1}$
Output: $s_0, \cdots, s_{n-1}$

1. Pre-computation:
   - For $j \in \Gamma(d)$, compute A_j and $g_{\alpha^j}(x)$ if $A_j \neq 0$.
   - Randomly select k with $\gcd(k, 2^n - 1) = 1$, and compute the polynomial $p_k(x) = \prod_{j \neq k: A_j \neq 0, j \in \Gamma(d)} g_{\alpha^j}(x)$.

   The steps 2-6 are the same as in Procedure 1.

8 Discussion

If we do not know D (or L) consecutive bits of $\{b_i\}$, then, consequently, we do not have n consecutive bits of $\{u_i\}$. However, if we know $(u_{k_0}, u_{k_1}, \cdots, u_{k_{n-1}})$ from some known segments of $\mathbf{b}$, then the matrix M in Section 5 becomes $M = (m_{ij})$ where $m_{ij} = \alpha^{k_i 2^j}$ which may not be a Vandermonde matrix. In order to have the linear subspace attack to work when $detM \neq 0$, we can have an extractor for retrieving β, and therefore, the initial state of $\mathbf{s}$. So, the problem becomes that how many bits we actually need to form an extractor with nonzero determinant of M. The method developed here can be also be applied to the case of combinatorial generators. We will discuss it in a separate paper.

9 Acknowledgements

This work was supported by the Norwegian Research Council.

References

1. S.W. Golomb and G. Gong, *Signal Design with Good Correlation: for Wireless Communications, Cryptography and Radar Applications*, Cambridge University Press, 2005.
2. G. Gong, Analysis and Synthesis of Phases and Linear Complexity of Non-Linear Feedforward Sequences, *Ph.D. thesis*, University of Elec. Sci. and Tech. of China, 1990.
3. P. Hawkes and G. Rose, "Rewriting variables: The complexity of fast algebraic attacks on stream ciphers," Advances in Cryptology - Crypto 2004 (Ed: Matt Franklin), Springer Lecture Notes in Computer Science, LNCS, vol. 3152, pp. 390-406, 2004.
4. T. Herlestam, On Functions of Linear Shift Register Sequences, *Advances in Cryptology – Eurocrypt 85'*, Lecture Notes in Computer Science, LNCS 0219, pp. 119-129, Springer-Verlag, 1985.
5. K.G. Paterson, Root Counting, the DFT and the Linear Complexity of Nonlinear Filtering, *Designs, Codes and Cryptography*, Vol. 14 (1998), 247–259.
6. S. Rønjom and T. Helleseth, A New Attack on the Filter Generator, *IEEE Transactions on Information Theory*, vol. 53, no. 5, pp. 1752-1758, 2007.
7. R.A. Rueppel, *Analysis and Design of Stream Ciphers*, Springer-Verlag, 1986.