On Algorithms for the k-Error Linear Complexity of Sequences over $GF(p^m)$ with Period p^n

Takayasu Kaida

Corrections

page	line	error		correction
Abstract	L8	with period p^n.	$\longrightarrow$	with period 2^n.
p.4	L10	of $\vec{e}$ do not	$\longrightarrow$	of $\{e_i\}$ does not
p.7	L13	over $GF(3^3)$	$\longrightarrow$	over $GF(3)$
	L-10	two algorithm	$\longrightarrow$	two algorithms
	L-8	these algorithm is	$\longrightarrow$	these algorithms are
p.35	L2	by the author et al.	$\longrightarrow$	by Imamura and Moriuchi
p.43	L-11	of Proposition 1	$\longrightarrow$	of Proposition 4.1
p.60	L-1	the discreet Fourier	$\longrightarrow$	the discrete Fourier
p.65	L1	$q \times M$ the change	$\longrightarrow$	$q \times p^n$ the change
p.68	L7	nearly 2/3 times	$\longrightarrow$	nearly 1/2 times
p.68	L-10	It become	$\longrightarrow$	It becomes
P.70	L-10	rithm have same	$\longrightarrow$	rithms have same
p.70	L-5	nearly 2/3 to	$\longrightarrow$	nearly 1/2 to
p.71	L-12	modifications of	$\longrightarrow$	modification of

Abstract

The linear complexity (LC) of a sequence has been used as a convenient measure of unpredictability of the sequence. However it is known that the LC has such an unnatural property as an extreme increase by one-symbol substitution, one-symbol insertion or one-symbol deletion. The k-error linear complexity (k-LC) defined by M. Stamp et al. is very effective for reducing the above-mentioned unnatural property of the LC. M. Stamp et al. proposed the fast algorithm for the k-LC of sequences over $GF(2)$ with period p^n, named as the Stamp-Martin algorithm. The Stamp-Martin algorithm is derived from the Games-Chan algorithm for the LC of binary sequences with period 2^n by the cost vector whose element means the minimum number of changes about the original sequence such that the k-LC does not increase and previous conditions are kept. The main result in this dissertation is the algorithm for the k-LC and the error vector which gives the k-LC of sequences over $GF(p^m)$ with period p^n, where p is a prime and n and m are positive integers. This generalization is derived by the generalized Games-Chan algorithm for the LC of sequences over $GF(p^m)$ with period p^n. Firstly another algorithm of the k-LC of sequences over $GF(2)$ with period 2^n that is different from the Stamp-Martin algorithm is derived. Moreover this algorithm gives not only the k-LC but also an error vector. It is shown by the shift and offset of the cost that this algorithm is equivalent to the Stamp-Martin algorithm. Next in order to generalize the Stamp-Martin algorithm into non-binary sequences, I discuss sequences over $GF(3)$ with period 3^n. The algorithm for the k-LC and the error vector of sequences over $GF(3)$ with period 3^n is derived. Hence the two algorithms for the k-LC and the error vector of sequences over $GF(p^m)$ with period p^n is shown by the generalized Games-Chan algorithm. One of them, called the generalized k-LC algorithm, does not use the shift and offset of the cost and consists two steps as the step of computing the k-LC and the step of computing the error vector. Other one, called the generalized Stamp-Martin algorithm, use the shift and offset of the cost and consists only one step given the k-LC and the error vector at the same time. At the end of each chapter numerical examples of proposed algorithms in that chapter are given.

Acknowledgment

I would like to express my sincere appreciation and thanks to Professor Kyoki Imamura with Kyushu Institute of Technology, the chairman of the committee for this dissertation, for introducing several research problems and giving his time to discus about these problems and encouraging by many advice and suggesting to me that I try to work as a research associate at Yatsushiro National College of Technology.

I would like to express my sincere appreciation to Professor Takeshi Shinohara, Professor Tsutomu Sasao and Professor Yuji Oie with Kyushu Institute of Technology, the other members of the committee for this dissertation. This committee gives useful advice for improving this dissertation and my public hearing.

I would like to express my sincere appreciation and thanks to Professor Tsutomu Moriuchi with Yatsushiro National College of Technology for introducing my work as a research associate at Yatsushiro National College of Technology and giving many comments and advice about research and work.

I would like to express thanks to Doctor Satoshi Uehara with the research associate of Kyushu Institute of Technology for many discussions, advice and helps.

I would like to express thanks to Ms. Cai Mian for giving many discussions and comments in my graduate school days

Finally, I would like to express sincere appreciation and thanks to many persons who gave many comments and helps, especially the members of Imamura laboratory at Kyushu Institute of Technology and the stuffs of Yatsushiro National College of Technology.

On Algorithms for the k-Error Linear Complexity of Sequences over $GF(p^m)$ with Period p^n

Takayasu Kaida

Contents

Chapter 1

Introduction

1.1 Background

The linear complexity (LC) of a sequence has been used as a convenient measure of unpredictability of the sequence, i.e., difficulty in recovering more of the sequence from a short captured segment, where the LC of a sequence is defined as the length of the shortest LFSR (linear feedback shift register) that generates the sequence[Rup86]. The LC of a sequence

$$\{a_i\} = (a_0, a_1, \cdots) \tag{1.1}$$

over $GF(q)$, denoted as LC($\{a_i\}$), is defined as the minimum L satisfying that fixed c_0, c_1, $\cdots$, c_{L-1} exist in $GF(q)$ such that

$$a_{L+i} + c_{L+i-1}a_{L+i-1} + \cdots + c_0 a_i = 0 \tag{1.2}$$

for $i \geq 0$. There are many useful algorithms such as the Berlekamp-Massey algorithm[Berl68, Mas69], the method using discrete Fourier transform[MS86] and the method of continuous fraction[Mil75]. Therefore the LC is studied by many researcher and there are many results about the LC. However it is known that the LC has such an unnatural property as an extreme increase or decrease by one-symbol substitution[IMU91, DI98], one-symbol insertion[UI96] and one-symbol deletion[UI97].

In 1993 M. Stamp and C. F. Martin[SM93] defined the k-error LC (k-LC) of periodic sequences as the smallest LC that can be obtained when any k or fewer of the symbols of a sequence are altered in one period. The k-LC is very effective for reducing the above-mentioned unnatural property of the LC. The k-LC of sequences is a very natural and useful generalization of the LC. The k-LC of a sequence $\{a_i\}$ over $GF(q)$ with period N is defined as

$$k\text{-}LC = \min\{LC(\{a_i + e_i\}) | w_H(\vec{e}) \leq k\}, \qquad (1.3)$$

where $\{e_i\} = (e_0, e_1, \cdots)$ is a sequence over $GF(q)$ with period N and $w_H(\vec{e})$ is the Hamming weight of a vector $\vec{e} = (e_0, e_1, \cdots, e_{N-1})$ with length N, especially the vector $\vec{e}$ which gives k-LC is called the error vector. Note that the period of $\vec{e}$ do not need the minimum period, i.e., it is sufficient to satisfy

$$e_{N+i} = e_i \quad \text{for} \quad i \geq 0. \qquad (1.4)$$

The sphere complexity defined by C. Ding, G. Xiao and W. Shen in 1991 [DXS91] is essentially the same as k-LC except to don't care about the LC of the input sequence. The definition of the sphere complexity of a sequence $\{a_i\}$, denoted as SC($\{a_i\}$) with period N, is

$$SC(\{a_i\}) = \min\{LC(\{a_i + e_i\}) | 1 \leq w_H(\vec{e}) \leq k\}. \qquad (1.5)$$

I must note that the sphere complexity defined by Ding, Xiao and Shan in their book [DXS91] is earlier than the k-LC and they are essentially the same (but not completely the same). In this dissertation will use the k-LC instead of the sphere complexity in spite of the earlier introduction of the sphere complexity because of the following two reasons. Firstly the use of the term "LC" is desirable to show that these complexities are natural generalizations of the LC. Secondly an efficient algorithm has been given only for the k-LC of a binary sequence with period 2^n. By the way the k-LC and the sphere complexity are not completely the same, since for an m-sequence over $GF(q)$ with period $N = q^n - 1$ the 1-LC is equal to the LC of an m-sequence valued n, but the 1-sphere complexity is equal to $N - n$[DXS91].

Unfortunately an effective algorithm for computing the k-LC has been known only for sequences over $GF(2)$ with period 2^n, named the Stamp-Martin algorithm[SM93]. The Stamp-Martin algorithm uses the Games-Chan algorithm[GC83] for computing the LC of sequences over $GF(2)$ with period 2^n. In 1991 by C. Ding, G. Xiao and W. Shen[DXS91] the generalized Games-Chan algorithm of sequences over $GF(p^m)$ with period p^n is proposed and independently the same algorithm proposed by K. Imamura and T. Moriuchi[IM93] in 1993.

The Stamp-Martin algorithm is very fast but this algorithm gives only the value of the k-LC. I think not only the k-LC but also the error vector of a sequence is very important in view of applications such as the field of computer science, communication systems and cryptography.

Recently the author et al.[KUI96, KUI96-2, KUI98, KUI98-2, KUI99] gave algorithms for the k-LC of sequences over $GF(p^m)$ with period p^n, p a prime.

Firstly another algorithm for the k-LC of sequences over $GF(2)$ with period 2^n was proposed by the author et al.[KUI96-2] using the shift and offset of the cost. Moreover equivalent between the Stamp-Martin algorithm and the proposed algorithm was showed and modified algorithms from the Stamp-Martin algorithm and the proposed algorithm were proposed, respectively in order to decide one of the error vectors in [KUI96-2].

Secondly the author et al.[KUI96] showed the algorithm for the k-LC of sequences over $GF(3)$ with period 3^n by generalization of the cost vector into the cost matrix in 1996. At the same time this algorithm derives one of error vectors which gives the k-LC of the sequence. Since the algorithm uses the generalized version of the Games-Chan algorithm for computing the LC of sequences over $GF(3)$ with period 3^n, more generalization of the algorithm into sequences over $GF(p^m)$ with period p^n is possible by using the generalized Games-Chan algorithm for the LC of sequences over $GF(p^m)$ with period p^n.

Finally the author et al.[KUI99, KUI98-2] proposed two generalized algorithm for k-LC and the error vector of sequences over $GF(p^m)$ with period p^n. One of them is the true generalization of the Stamp-Martin algorithm by using the concepts called the shift and offset of the cost[KUI98-2]. This algorithm gives the k-LC and an error vector at the same time in step by step. Another algorithm remains about the error vector throughout all the steps for computing the k-LC but stores data to determine an error vector. After k-LC is decided the algorithm calculate an error vector by reading the stored data in the reverse order from the last step to the first step.

1.2 In This Dissertation

In This Dissertation I show the generalized algorithms for the k-LC and the error vector of sequences over $GF(p^m)$ with period p^n, where p is a prime and n and m are positive integers.

Firstly the algorithm for the k-LC and the error vector of sequences over $GF(2)$ with period 2^n is shown in chapter 2. I describe the Games-Chan algorithm for the LC and the Stamp-Martin algorithm for the k-LC of sequence over $GF(2)$ with period 2^n This algorithm is different from the Stamp-Martin algorithm about do not use the shift and offset of the cost and the cost matrix not but the cost vector in the Stamp-Martin algorithm. I show that the Stamp-Martin algorithm is derived from the proposed algorithm by the shift and offset of the cost. Moreover the proposed algorithm can derive an error vector which gives the k-LC. After the algorithm for the k-LC is over, the algorithm for the error vector uses memories of the change value at the algorithm for k-LC.

In Chapter 3, I propose the algorithm over $GF(3)$ with period 3^n in order to generalize the Stamp-Martin algorithm into non-binary sequences. Since the concepts called the shift and offset of the cost is applied, I can generalize the Stamp-Martin algorithm into sequences over $GF(3)$ with period 3^m at the first step for generalization into sequences over $GF(p^m)$ with period p^n. Hence I show the generalized Games-Chan algorithm for the LC into sequences over $GF(3)$ with period 3^n. Using this algorithm the generalization of the algorithm for the k-LC is derived and the algorithm for the error vector is given originally. At the end of Chapter 3 an example, denoted as Example 3.1, of the performance of the proposed algorithm for the k-LC and the error vector and the profile of the value k-LC about k are given.

Moreover I propose more generalized algorithm for the k-LC and the error vector into sequences over $GF(p^m)$ with period p^n, where p is a prime, in Chapter 4. Firstly the generalized Games-Chan algorithm is described as a preparation. Then I propose the generalized algorithm for the k-LC and the error vector into sequences over $GF(p^m)$ with period p^n. This algorithm has also two parts as the algorithm for the k-LC and the algorithm for the error vector. The algorithm for the error vector executes with the memories of the change value after the algorithm for k-LC. Two examples are shown at the end of Chapter 4. In Example 4.1 the performance of the proposed algorithm for the k-LC and the error vector of sequences over $GF(3)$ with period 3^3. In Example 4.2 the performance of the proposed algorithm for the k-LC and the error vector of sequences over $GF(3^3)$ with period 3^3 also.

In Chapter 5 I propose the another algorithm for the k-LC and the error vector of sequences over $GF(p^m)$ with period p^n. This algorithm gives the k-LC and the error vector at the same time. Firstly several preparations of the cost matrices of the vector $\vec{a}$ and the vector $\vec{b}$ and the shift and offset of the cost and so on, is described. Next I show the proposed algorithm for the k-LC of sequences over $GF(p^m)$ with period p^n and the algorithm for the error vector of sequences over $GF(p^m)$ with period p^n. This algorithm for the error vector is different from the algorithm proposed in Chapter 4 about only one part for the k-LC and the error vector not but two parts separated the k-LC and the error vector. A numerical example, denoted as Example 5.1, is the performance of the algorithm for the k-LC and the error vector at the same time.

Finally I discuss the computational complexity in order to compare two algorithm for the k-LC and the error vector of sequences over $GF(p^m)$ with period p^n in Chapter 4 and Chapter 5. After these algorithm is written again, the time-complexity and the space-complexity are evaluated. The time-complexity is the number of the addition and subtraction except the comparison and the setting variable in this dissertation. The space-complexity is the number of the required memories in these algorithms.

In chapter 7, I describe the conclusions and the future works. The result of this work is explained in this chapter. I need discuss the problem of the period. More works in future are found in this chapter.

Chapter 2

Algorithms for the k-LC over $GF(2)$ with Period 2^n

2.1 Introduction

In this chapter an alternative derivation of the Stamp-Martin algorithm is given. This method can compute not only k-LC but also an error vector with Hamming weight $\leq k$ which gives the k-LC.

Unfortunately the Stamp-Martin algorithm[SM93] can apply only for sequences over $GF(2)$ with period 2^n, because of using the Games-Chan algorithm[GC83] for computing the LC of sequences over $GF(2)$ with period 2^n.

In next Section 2.2 I show algorithm for computing the k-LC of sequences over $GF(2)$ with period 2^n similar to the Stamp-Martin algorithm after the Games-Chan algorithm for the LC of sequences over $GF(2)$ with period 2^n is given.

Secondly I propose another algorithm for the k-LC of sequences over $GF(2)$ with period 2^n, denoted as the Algorithm I in Section 2.3. By the extended cost matrix from the cost vector at the Stamp-Martin algorithm, the Algorithm I is derived. And the Algorithm I is executed without the shift and offset of the cost matrix. Therefore it is possible that the value of k is fixed and the vector $\vec{a}$ is given simply at each step. Moreover I propose the algorithm for the error vector in the Algorithm I by using the stored data about the value of changing at each step. The validity of the Algorithm I is shown by the property of the Algorithm I as Proposition 2.1. An example of the Algorithm I, denoted as Example 2.1, is given in the end of this section.

In Section 2.4 I try to derive the Stamp-Martin algorithm from the Algorithm I proposed in Section 2.2 by rewriting the Algorithm I into the Algorithm II by using the shift and offset of the cost matrix. It is clear to be equivalent between the Algorithm I and the Stamp-Martin algorithm.

I propose the part of computing an error vector of Hamming weight $\leq k$ which gives the k-LC about the Algorithm I in Section 2.5. An efficient computation is necessary not only of the k-LC but also of an error vector, since the number of possible candidates of the error vector

$$\sum_{0 \leq i \leq k} \binom{2^n}{i} \tag{2.1}$$

is very large number even for moderate n and k.

At the end of this chapter an example, denoted Example 2.2, of the performance of Algorithm II is given.

2.2 The Stamp-Martin Algorithm

In this section I briefly review the Games-Chan algorithm[GC83] and the Stamp-Martin algorithm[SM93].

In this chapter I will consider a periodic sequence

$$\{a_i\} = \{a_0, a_1, a_2, \cdots\} \tag{2.2}$$

over $GF(2)$ with period $N = 2^n$ for $n \geq 1$. Let

$$\vec{a} = (a_0, a_1, \cdots, a_{N-1}). \tag{2.3}$$

I will write $\vec{a}$ as

$$\vec{a} = (\vec{a}(0), \vec{a}(1)), \tag{2.4}$$

where

$$\left.\begin{array}{rcl} \vec{a}(0) & = & (a_0, a_1, \cdots, a_{N/2-1}), \\ \vec{a}(1) & = & (a_{N/2}, a_{N/2+1}, \cdots, a_{N-1}). \end{array}\right\} \tag{2.5}$$

For a binary periodic sequence (2.2) there exists the following fast algorithm[GC83] for computing the LC of $\{a_i\}$ denoted as LC($\{a_i\}$).

[Games-Chan Algorithm]

```
1. Initial values:
   $N = 2^n$,   $LC = 0$,   $\vec{a} = Eq.(2.3)$.
2. Repeat the following (a)~(c) until $N = 1$.
   (a) From the given $\vec{a}$ in (2.4), compute
       $\vec{b} = (b_0, b_1, \cdots, b_{N/2-1})$ by
           $\vec{b} = \vec{a}(0) + \vec{a}(1).$          (2.6)
   (b) If $\vec{b} = \vec{0}$, then
           $\vec{a} \leftarrow \vec{a}(0), \quad N \leftarrow N/2$
       and go to (a).
   (c) If $\vec{b} \neq \vec{0}$, then
           $\vec{a} \leftarrow \vec{b}, \quad LC \leftarrow LC + N/2, \quad N \leftarrow N/2$
       and go to (a).
3. If $\vec{a} \neq 0$ for $N = 1$, then
           $LC \leftarrow LC + 1.$
```

The final LC is equal to the LC($\{a_i\}$).

The basic logic of the Stamp-Martin algorithm[SM93] for computing the k-LC of a binary sequence (2.2) with period $N = 2^n$, denoted as k-LC($\{a_i\}$), is "apply the Games-Chan algorithm, but if $\vec{b} \neq \vec{0}$ and I can force $\vec{b} = \vec{0}$, I do so". The algorithm uses the cost of current element a_i, denoted as $A(i)$, as a measure of the "cost" (in terms of the number of bit changes required in the original sequence (2.3)) of changing the current element a_i without disturbing the results of any previous steps. The algorithm can be summarized as follows (I use here some notations different from those used in [SM93], so that the comparison with our method stated in the next section will become more obvious).

[Stamp-Martin Algorithm]

1. Initial values:
   $N = 2^n$, $LC = 0$, $\vec{a} = Eq.(2.3)$.
   $A(0) = A(1) = \cdots = A(N-1) = 1$.

2. Repeat the following (a)~(c) until $N = 1$.

   (a) From the given $\vec{a}$ in (2.4), compute $\vec{b} = (b_0, b_1, \cdots, b_{N/2-1})$ by

   $$\vec{b} = \vec{a}(0) + \vec{a}(1). \tag{2.7}$$

   Compute TB by

   $$TB = \sum_{0 \le i \le N/2-1} b_i \min(A(i), A(i+N/2))$$

   (b) If $TB \le k$, then firstly

   $$k \leftarrow k - TB; \quad N \leftarrow N/2$$

   and secondly compute $\vec{A} = (A(0), A(1), \cdots, A(N-1))$ and $\vec{a} = (a_0, a_1, \cdots, a_{N-1})$ by
   - if $b_i = 1$ and $A(i) \le A(i+N)$ then
   $$A(i) \leftarrow A(i+N) - A(i); \quad a_i \leftarrow a_i + 1$$
   - if $b_i = 1$ and $A(i) > A(i+N)$ then
   $$A(i) \leftarrow A(i) - A(i+N)$$
   - if $b_i = 0$ then
   $$A(i) \leftarrow A(i) + A(i+N)$$

   and thirdly

   $$\vec{a} \leftarrow (a_0, a_1, \cdots, a_{N-1}).$$

   If $N \ne 1$, then go to (a).

   (c) If $TB > k$, then firstly

   $$N \leftarrow N/2; \quad LC \leftarrow LC + N$$

   and secondly compute $\vec{A} = (A(0), A(1), \cdots, A(N-1))$ and $\vec{a} = (a_0, a_1, \cdots, a_{N-1})$ by

$$A(i) \leftarrow \min(A(i), A(i+N))$$

and

$$a_i \leftarrow a_i + a_{i+N}.$$

If $N \neq 1$, then go to (a).

3. If $\vec{a} \neq 0$ and $A(0) > k$ for $N = 1$, then

$$LC \leftarrow LC + 1.$$

The final LC is equal to the k-LC($\{a_i\}$).

2.3 Another Algorithm for Computing the k-LC

In this section I give a new method for computing the k-LC of binary sequence (2.2) with period 2^n by applying the method of [KUI96]. Some basic properties of the algorithm are also shown. The relation between our algorithm and the Stamp-Martin algorithm will be shown in the next section.

Instead of the cost vector $\vec{A} = (A(0), A(1), \cdots, A(N-1))$ in the Stamp-Martin algorithm I will use the following $2 \times N$ cost matrix $A = [A(j,i)]$, $(j = 0, 1;\ 0 \leq i \leq N-1,\ N = 2^{n-u})$ at the step u, and cost vector $\vec{B} = (B_0, B_1, \cdots, B_{N/2-1})$ of the vector $\vec{b}$ at the step u, $A(j,i)$ is defined as the minimum number of changes about the original sequence (2.3) with period $N = 2^n$ necessary and sufficient for changing the current element a_i to $a_i + j$ with the condition that all the previous $\vec{b} = \vec{0}$'s are kept unchanged. The definition of B_i is similar to that of $A(j,i)$ except that I change b_i to 0.

The Games-Chan algorithm for a binary sequence with period 2^n consists of $n+1$ steps and at the step u, $(0 \leq u \leq n)$, the length of the vector $\vec{a}$ is equal to 2^{n-u}. I need to find formulas for determining (i) the cost vector $\vec{B}$ at the step u and (ii) the cost matrix $A' = [A'(j,i)]$ at the step $u+1$ from the cost matrix $A = [A(j,i)]$ and the vector $\vec{b}$ at the step u.

Note that $b_i = a_i + a_{i+N/2}$ and B_i is the cost (in terms of the number of symbol changes required in the original sequence (2.3)) of making b_i to 0 with the condition that the previous $\vec{b} = \vec{0}$'s are kept unchanged. Making b_i to 0 with such a condition is performed by changing a_i and $a_{N/2+i}$ to $a_i + d_0$ and $a_{N/2+i} + d_1$, respectively, where d_0 and d_1 must satisfy $(a_i + d_0) + (a_{N/2+i} + d_1) = 0$. Therefore I have

$$B_i = \min_{j \in \{0,1\}} (A(j,i) + A(j + b_i, N/2 + i)) \tag{2.8}$$

Let $\vec{a'} = (a'_0, a'_1, \cdots, a'_{N/2-1})$ be the vector $\vec{a}$ at the step $u + 1$. There are two cases; (i) $a'_i = a_i$ if I can make $\vec{b} = \vec{0}$ at the step u and (ii) $a'_i = a_i + a_{N/2+i}$ otherwise. Note that $A'(j,i)$ is the cost of making a'_i to $a'_i + j$ with the condition that the previous $\vec{b} = \vec{0}$'s are unchanged. Therefore making a'_i to $a'_i + j$ must be performed in both (i) and (ii) by changing a_i and $a_{N/2+i}$ to $a_i + d_0$ and $a_{N/2+i} + d_1$, respectively, where d_0 and d_1 must satisfy $a_i + d_0 = a_i + j$ and $(a_i + d_0) + (a_{N/2+i} + d_1) = a_i + a_{N/2+i}$ in case of (i), and $(a_i + d_0) + (a_{N/2+i} + d_1) = (a_i + a_{N/2+i}) + j$ in case of (ii), respectively. Those two cases can be decided by using the total cost

$$TB = B_0 + B_1 + \cdots + B_{N/2-1} \tag{2.9}$$

at the step u, i.e., the case (i) corresponds to $TB \leq k$ and the case (ii) to $TB > k$. Therefore I have

$$A'(j,i) = A(j,i) + A(j + b_i, N/2 + i) \quad \text{if } TB \leq k, \tag{2.10}$$

$$A'(j,i) = \min_{s \in \{0,1\}} (A(s,i) + A(j + s, N/2 + i)) \quad \text{if } TB > k. \tag{2.11}$$

I will define a $2 \times N$ matrix $D = D(u) = \left[\vec{D}(j,i)\right]$, where $\vec{D}(j,i)$ is a binary 2-tuple, $j = 0, 1$, $0 \leq i \leq N - 1$ and $N = 2^{n-u}$. The element $\vec{D}(j,i) = (d_0, d_1)$ is defined as

$$A'(j,i) = A(d_0, i) + A(d_1, N/2 + i). \tag{2.12}$$

From (2.10)-(2.12) I have

$$d_0 = j, \quad d_1 = j + b_i, \qquad \text{if } TB \leq k, \tag{2.13}$$

$$d_0 = s, \quad d_1 = j + s, \qquad \text{if } TB > k, \tag{2.14}$$

where s is a solution of $A'(j,i) = A(s,i) + A(j+s, N/2+i)$ in (2.11). It may happen that the 2-tuple (d_0, d_1) in (2.14) is not unique because of non-uniqueness of s in (2.11).

I have the following algorithm for computing the k-LC.

[Algorithm I]

1. Initial values:
   $N = 2^n, \quad LC = 0, \quad \vec{a} = Eq.(2.3)$.
   $A(0,i) = 0, \quad A(1,i) = 1$ for $0 \le i \le N-1$

2. Repeat the following (a)~(c) until $N = 1$.

   (a) From the given $\vec{a}$ in (2.4), compute
   $\vec{b} = (b_0, b_1, \cdots, b_{N/2-1})$ by (2.7) and its cost vector $\vec{B} = (B_0, B_1, \cdots, B_{N/2-1})$ by (2.8) and TB by (2.9).

   (b) If $TB \le k$, then firstly
   • compute the cost matrix $A' = [A'(j,i)]$ and the matrix $D = \left[\vec{D}(j,i)\right]$, $(\vec{D}(j,i) = (d_0, d_1))$, at the next step by (2.10) and (2.13), respectively,
   and secondly
   • $N \leftarrow N/2, \quad \vec{a} \leftarrow \vec{a}(0), \quad A \leftarrow A'$.
   If $N \ne 1$, then go to (a).

   (c) If $TB > k$, then firstly
   • compute the cost matrix $A' = [A'(j,i)]$ and the matrix $D = \left[\vec{D}(j,i)\right]$, $(\vec{D}(j,i) = (d_0, d_1))$, at the next step by (2.11) and (2.14), respectively,
   and secondly
   • $N \leftarrow N/2, \quad \vec{a} \leftarrow \vec{a}(0) + \vec{a}(1), \quad A \leftarrow A', \quad LC \leftarrow LC + N$.
   If $N \ne 1$, then go to (a).

3. If $A(a_0, 0) > k$ for $N = 1$, then

$$LC \leftarrow LC + 1.$$

The final LC is equal to the k-LC($\{a_i\}$).

The validity of the Algorithm I can be proved by the following proposition.

Proposition 2.1

Let $MAS(u)$ at the step u, $(0 \leq u \leq n)$, be defined by

$$MAS(u) = \sum_{0 \leq i \leq N-1} \min(A(0,i), A(1,i)), \quad N = 2^{n-u}. \tag{2.15}$$

Let $TB(u)$ be TB at the step u. I have the following (i)-(iv).

(i) $MAS(u) \leq TB(u)$.

(ii) $MAS(u+1) = TB(u) \qquad$ if $TB(u) \leq k$.

(iii) $MAS(u+1) = MAS(u) \quad$ if $TB(u) > k$.

(iv) $0 = MAS(0) \leq MAS(1) \leq \cdots \leq MAS(n) \leq k$.

proof

(i) This is obvious from (2.8), (2.9) and (2.15).

(ii) If $TB(u) \leq k$, then $A'(j,i) = A(j,i) + A(j+b_i, N/2+i)$ from (2.12) and (2.14). I have

$$MAS(u+1) = \sum_{0 \leq i \leq N/2-1} \min(A'(0,i), A'(1,i)) = TB(u)$$

from (2.8).

(iii) This is also obvious from (2.11) and (2.15).

(iv) $MAS(0) = 0$ is trivial from the initial values. The remaining relation can be easily proven from (ii) and (iii) by the induction on u.

The validity of the Algorithm I is now clear, since its basic logic is the same as that of the Stamp-Martin algorithm and Proposition2.1 ensures that $\min(A(0,0), A(1,0)) \leq k$, which means that at least one of the minimum cost necessary and sufficient for changing a_0 at the step n to either a_0+0 or a_0+1 is not greater than k. In section 2.5 I will give a method for computing an error vector by reading the matrices $D(n)$, $D(n-1)$, $\cdots$, $D(1)$ in this order.

[**Example 2.1**]

Consider the 2-LC (i.e., $k = 2$) of the following binary sequence with period $32 = 2^5$ defined by

$$\vec{a} = (0001\ 1010\ 1001\ 1010 \quad 1000\ 1010\ 1001\ 1010). \qquad (2.16)$$

(Step 0) $A(0,i) = 0, \quad A(1,i) = 1, \quad \text{for } 0 \le i \le 31.$
$\vec{b} = (1001\ 0000\ 0000\ 0000), \quad \vec{B} = (1001\ 0000\ 0000\ 0000),$
$TB = 2 = k, \qquad LC = 0.$

(Step 1) $\vec{a} = (0001\ 1010 \quad 1001\ 1010),$
$A(0, 0 \le i \le 15) = (1001\ 0000 \quad 0000\ 0000),$
$A(1, 0 \le i \le 15) = (1221\ 2222 \quad 2222\ 2222).$
$\vec{D}(0,*) = (01,00,00,01,\ 00,00,00,00,\ 00,00,00,00,\ 00,00,00,00),$
$\vec{D}(1,*) = (10,11,11,10,\ 11,11,11,11,\ 11,11,11,11,\ 11,11,11,11),$
$\vec{b} = (1000\ 0000), \quad \vec{B} = (1001\ 0000), \quad TB = 2 = k, \quad LC = 0.$

(Step 2) $\vec{a} = (0001 \quad 1010),$
$A(0, 0 \le i \le 7) = (3001 \quad 0000),$
$A(1, 0 \le i \le 7) = (1443 \quad 4444),$
$\vec{D}(0, 0 \le i \le 7) = (01,00,00,00, \quad 00,00,00,00),$
$\vec{D}(1, 0 \le i \le 7) = (10,11,11,11, \quad 11,11,11,11),$
$\vec{b} = (10\ 11), \quad \vec{B} = (10\ 43), \quad TB = 8 > k = 2,$
$LC = 0 + 4 = 4.$

(Step 3) $\vec{a} = (10 \quad 11),$
$A(0, 0 \le i \le 3) = (3,0, \quad 0,1), \quad A(1, 0 \le i \le 3) = (1,4, \quad 4,3),$
$\vec{D}(0, 0 \le i \le 3) = (00,00, \quad 00,00), \quad \vec{D}(1, 0 \le i \le 3) = (10,01, \quad 01,10),$
$\vec{b} = (0\ 1), \quad \vec{B} = (3\ 3), \quad TB = 6 > k = 2, \quad LC = 4 + 2 = 6.$

(Step 4) $\vec{a} = (0 \quad 1), \quad A(0, 0 \le i \le 1) = (3,\ 1), \quad A(1, 0 \le i \le 1) = (1,\ 3),$
$\vec{D}(0, 0 \le i \le 1) = (00, \quad 00), \quad \vec{D}(1, 0 \le i \le 1) = (10, \quad 01),$
$\vec{b} = (1), \quad \vec{B} = (2), \quad TB = 2 = k, \quad LC = 6 + 0 = 6.$

(Step 5) $\vec{a} = (0), \quad A(0,0) = 6, \quad A(1,0) = 2,$
$\vec{D}(0,0) = (01), \quad \vec{D}(1,0) = (10), \quad A(0,0) = 6 > k = 2,$
$LC = 6 + 1 = 7.$

I have k-LC($\{a_i\}$) $= 7$.

2.4 Derivation of the Stamp-Martin Algorithm

In this section I will show that the Stamp-Martin algorithm can be obtained from the Algorithm I by making

$$A(0,0) = A(0,1) = \cdots = A(0,N-1) = 0, \quad N = 2^{n-u} \tag{2.17}$$

at each step u, $(0 \le u \le n)$.

Let $\vec{a} = (a_0, a_1, \cdots, a_{N-1})$ and $A = [A(j,i)]$ be the vector $\vec{a}$ and the cost matrix at the step u. Let $\vec{a'} = (a'_0, a'_1, \cdots, a'_{N/2-1})$ and $A' = [A'(j,i)]$ be the vector $\vec{a}$ and the cost matrix at the step $u+1$. Assume that (2.17) holds until at the step u. In order to make (2.17) true at step $u+1$, I introduce two operations, i.e., "shift" and "offset" as follows. The "shift" is used to make $0 \le A'(0,i) \le A'(1,i)$ when $0 \le A'(1,i) \le A'(0,i)$ (inclusion of "=" is arbitrary) by $a'_i \leftarrow a_i + 1$, $A'(0,i) \leftarrow A'(1,i)$ and $A'(1,i) \leftarrow A'(0,i)$. The "offset" is used to make $A'(0,i) = 0$ when $0 \le A'(0,i) \le A'(1,i)$ by $A'(0,i) \leftarrow A'(0,i) - A'(0,i)$, $A'(1,i) \leftarrow A'(1,i) - A'(0,i)$ and $k \leftarrow k - A'(0,i)$. Assume that (2.17) holds at step u. Then (2.8) becomes

$$B_i = \begin{cases} 0 & \text{if } b_i = 0, \\ \min(A(1,i), A(1,N/2+i)) & \text{if } b_i = 1. \end{cases} \tag{2.18}$$

First consider the case $TB \le k$.

- If $b_i = 1$ and $A(1,i) \le A(1,N/2+i)$, then from (2.10) I have $0 \le A'(1,i) = A(1,i) \le A(1,N/2+i) = A'(0,i)$ and I need in general both of "shift" and "offset", i.e., I make a change of $A'(1,i) \leftarrow A(1,N/2+i) - A(1,i)$, $a'_i \leftarrow a_i + 1$ and $k \leftarrow k - A(1,N/2+i)$.

- If $b_i = 1$ and $A(1,i) > A(1,N/2+i)$, then from (2.10) I have $A'(0,i) = A(1,N/2+i)$ and $A'(1,i) = A(1,i)$. In this case "shift" is not necessary but I may need "offset", i.e., I make a change of $A'(1,i) \leftarrow A(1,i) - A(1,N/2+i)$ and $k \leftarrow k - A(1,N/2+i)$.

- If $b_i = 0$, then I have $A'(0,i) = 0$ and $A'(1,i) = A(1,i) + A(1,N/2+i)$ from (2.10).

From (2.18) the reduction of k is in total $k \leftarrow k - TB$ and notice that $k - TB \geq 0$ by Proposition 2.1.

Next consider the case $TB > k$.

- From (2.11) I have $A'(0,i) = 0$ and $A'(1,i) = \min(A(1,i), A(1,N/2+i))$.

If $A(0,0) = 0$ is assumed in case of $N = 1$, then the increase of LC happens if and only if $a_0 = 1$ and $A(1,0) > k$.

Above discussion gives the following modified version of the Algorithm I.

[**Algorithm II**]

```
1. Initial values:
   $N = 2^n$,   $LC = 0$,   $\vec{a} = Eq.(2.3)$.
   $A(1,0) = A(1,1) = \cdots = A(1,N-1) = 1$.
2. Repeat the following (a)~(c) until $N = 1$.
   (a) From the given $\vec{a}$ in (2.4), compute
       $\vec{b} = (b_0, b_1, \cdots, b_{N/2-1})$ by (2.9), and the cost vector $\vec{B} = (B_0, B_1, \cdots, B_{N/2-1})$ by (2.18) and $TB$ by (2.9).
   (b) If $TB \leq k$, then firstly
           $k \leftarrow k - TB; \quad N \leftarrow N/2$
       and secondly compute $\vec{A} = (A(1,0), A(1,1), \cdots, A(1,N-1))$ and $\vec{a} = (a_0, a_1, \cdots, a_{N-1})$ by
       • if $b_i = 1$ and $A(i) \leq A(i+N)$ then
           $a_i \leftarrow a_i + 1; \quad A(i) \leftarrow A(i+N) - A(i)$
       • if $b_i = 1$ and $A(i) > A(i+N)$ then
           $A(i) \leftarrow A(i) - A(i+N)$
       • if $b_i = 0$ then
```

$$A(i) \leftarrow A(i) + A(i+N)$$

and thirdly

$$\vec{a} \leftarrow (a_0, a_1, \cdots, a_{N-1}).$$

If $N \neq 1$, then go to (a).

(c) If $TB > k$, then firstly

$$N \leftarrow N/2; \quad LC \leftarrow LC + N$$

and secondly compute $\vec{A} = (A(0), A(1), \cdots, A(N-1))$ and $\vec{a} = (a_0, a_1, \cdots, a_{N-1})$ by

$$A(i) \leftarrow \min(A(i), A(i+N)),$$
$$a_i \leftarrow a_i + a_{i+N}.$$

If $N \neq 1$, then go to (a).

3. If $\vec{a} \neq 0$ and $A(1,0) > k$ for $N = 1$, then

$$LC \leftarrow LC + 1.$$

The final LC is equal to the k-LC($\{a_i\}$).

It is clear that Algorithm II is the same as the Stamp-Martin algorithm.

2.5 Computation of an Error Vector

It is also important to find an error vector effectively together with the k-LC. Algorithm I is very suitable to find an error vector

$$\vec{e} = (e_0, e_1, \cdots, e_{N-1}), \quad N = 2^n, \tag{2.19}$$

which satisfies that (i) $\sum_{0 \leq i \leq N-1} e_i \leq k$ and (ii) the LC of the binary sequence corresponding to $\vec{a} + \vec{e}$ instead of the vector $\vec{a}$ is equal to k-LC($\{a_i\}$).

After finishing the Algorithm I, I can compute an error vector (2.19) by reading the matrices $D(u)$'s, $(1 \leq u \leq n)$, in the order from $u = n$ to $u = 1$ in the following way. I use the following vector

$$\vec{d}(u) = (\vec{d}_0, \vec{d}_1, \cdots, \vec{d}_{N-1}), \quad N = 2^{n-u}, \tag{2.20}$$

where $\vec{d_i} = (d_{0,i}, d_{1,i})$ is equal to either $\vec{D}(0,i)$ or $\vec{D}(1,i)$ with $\vec{D}(j,i)$ being the (j,i)-th element of the matrix $D(u)$ at the step u.

[Computation of an Error Vector]

1. Initial values:
   Start from $u = n$ and $N = 2^{n-u} = 1$. Let $\vec{a} = (s)$ be the vector $\vec{a}$ at step n. Let $A(j,0)$ be the element of the matrix A at step n. Let $\vec{D}(0,0)$ and $\vec{D}(1,0)$ be the elements of the matrix $D(n)$ at step n. Compute $\vec{d}(n) = (d_0, d_1)$ by $\vec{d}(n) = \vec{D}(s,0)$ if $A(s,0) \le k$ and $\vec{d}(n) = \vec{D}(s+1,0)$ if $A(s,0) > k$.
2. Repeat the following (a)~(b) until $u = 1$ and $N = 2^{n-1}$.
   (a) Given the vector $\vec{d}(u)$ of (2.20) with $\vec{d_i} = (d_{0,i}, d_{1,i})$. Also given the matrix $D'(u-1) = [D'(j,i)]$, $(j = 0,1;\ 0 \le i \le 2N-1)$.
   Compute vector $\vec{d'}(u-1) = (\vec{d'_0}, \vec{d'_1}, \cdots, \vec{d'_{2N-1}})$ with $\vec{d'_i} = (d'_{0,i}, d'_{1,i})$ by
   $$\vec{d'_i} = \vec{D}'(d_{0,i}, i), \quad \vec{d'_{N+i}} = \vec{D}'(d_{1,i}, N+i), \quad (0 \le i \le N-1). \tag{2.21}$$
   (b) $N \leftarrow 2N$, $u \leftarrow u-1$
   If $u \ne 1$ and $N \ne 2^{n-1}$, then go to (a).
3. If $u = 1$ and $N = 2^{n-1}$, write
   $$\vec{d}(1) = ((d_{0,0}, d_{1,0}), (d_{0,1}, d_{1,1}), \cdots, (d_{0,N-1}, d_{1,N-1})) \tag{2.22}$$
   Compute an error vector (2.19) by
   $$e_i = d_{0,i}, \quad e_{N+i} = d_{1,i}, \quad (0 \le i \le N-1). \tag{2.23}$$

[Example 2.2]
Consider the sequence (2.16) in Example 2.1 and $k = 2$. All the necessary data are given by Example 2.1. This computation runs as follows.

(Step 5) $\vec{d}(5) = (1,0)$, since $\vec{a} = (0)$, $A(0,0) = 6 > k = 2$, $A(1,0) = 2$, and
$\vec{D}(1,0) = (1,0)$

(Step 4) $\vec{d}(4) = (\vec{d_0}, \vec{d_1}) = ((1,0),(0,0))$, since from the matrix $D(4)$
in Example 1 and $\vec{d}(5)$, I have $\vec{d_0} = \vec{D}'(1,0) = (1,0)$ and $\vec{d_1} = \vec{D}'(0,1) = (0,0)$

(Step 3) $\vec{d}(3) = ((1,0),(0,0),(0,0),(0,0))$.

(Step 2) $\vec{d}(2) = ((1,0),(0,0),(0,0),(0,0),(0,0),(0,0),(0,0),(0,0))$.

(Step 1) $\vec{d}(1) = ((1,0),(0,0),(0,0),(0,1),(0,0),(0,0),(0,0),(0,0),(0,0),(0,0),(0,0),(0,0),(0,0),(0,0),(0,0),(0,0))$.

I have an error vector

$$\vec{e} = (1000\ 0000\ 0000\ 0000\ 0001\ 0000\ 0000\ 0000), \qquad (2.24)$$

since I have $e_0 = e_{19} = 1$ from $\vec{d}(1)$.

2.6 Conclusion

Another method for computing the k-LC of a binary sequence with period 2^n was given (Algorithm I in section 2.3).

An alternative derivation of the Stamp-Martin algorithm was shown to be possible from the Algorithm I (Algorithm II in section 2.4).

Computation of an error vector after finding the k-LC was given by using the Algorithm I. The similar computation is possible by using the Algorithm II if I include the matrix $D(u)$ with suitable modifications corresponding to the shifts. I have a conjecture about the computation of an error vector using the Stamp-Martin algorithm which does not use the matrices $D(u)$'s.

Since the Games-Chan algorithm was generalized to the non-binary sequences, the Stamp-Martin algorithm for binary sequences can be generalized to the one for sequences over $GF(p^m)$ with period p^n, where p is an odd prime. Such a generalization is truly possible as shown by the authors[KUI96].

Chapter 3

An Algorithm for the k-LC over $GF(3)$ with period 3^n

3.1 Introduction

Unfortunately an effective algorithm for computing the k-LC has been known only for sequences over $GF(2)$ with period 2^n, which is called the Stamp-Martin algorithm[SM93]. The Stamp-Martin algorithm uses the Games-Chan algorithm [GC83] for computing the LC of sequences over $GF(2)$ with period 2^n. Since the Games-Chan algorithm was generalized to the sequences over $GF(p^m)$ with period p^n, p a prime, by Ding, Xiao, and Shan[DXS91] and also by Imamura and Moriuchi[IM93], it seems to be possible to find a similar algorithm for the k-LC of sequences over $GF(p^m)$ with period p^n. In this chapter I propose an algorithm for the k-LC of sequences over $GF(3)$ with period 3^n, i.e. in case of $p = 3$ and $m = 1$, as the first step for generalization of the Stamp-Martin algorithm into sequences $GF(p^m)$ with period p^n. The algorithm is derived by the generalized Games-Chan algorithm for the LC of sequences over $GF(3)$ with period 3^n and using the modified cost not the same as that used in the Stamp-Martin algorithm.

In Section 3.2 I explain the generalized Games-Chan algorithm for the LC of sequences over $GF(3)$ with period 3^n, since this algorithm is applied in derivation of the proposed algorithm.

Secondly after several preparations of the cost matrices about the vector $\vec{a}$ and the vector $\vec{b}$, I derive an algorithm for the k-LC of sequences over $GF(3)$ with period 3^n. Moreover the property of the algorithm is shown as Proposition 3.1 in the end of Section 3.3.

In Section 3.4 I give a method for computing an error vector, which gives the k-LC, of sequences over $GF(3)$ with period 3^n. This algorithm uses the stored data about the value of changing at the algorithm for the k-LC after the k-LC is decided. It is easy in principle to generalize the method for sequences over $GF(p^m)$ with period p^n.

Finally I show an example, denoted as Example 3.1, of sequences over $GF(3)$ with period 3^n. This example consists of the performance of the proposed algorithm and the profile of the k-LC about the value k in Section 3.5.

3.2 Generalization of the Games-Chan Algorithm into $GF(3)$ with Period 3^n

In this chapter I consider only case that given sequences are over $GF(3)$ with period 3^n, since I want to show an algorithm for the k-LC of sequences over $GF(3)$ with period 3^n.

Let $\{a_i\} = \{a_0, a_1, a_2, \cdots\}$ be a sequence over $GF(3)$ with period $N = 3^n$ and

$$\vec{a} = (a_0, a_1, \cdots, a_{N-1}). \tag{3.1}$$

I will write $\vec{a}$ as

$$\vec{a} = (\vec{a}(0), \vec{a}(1), \vec{a}(2)), \tag{3.2}$$

where

$$\left.\begin{array}{rcl} \vec{a}(0) &=& (a_0, a_1, \cdots, a_{M-1}), \\ \vec{a}(1) &=& (a_M, a_{M+1}, \cdots, a_{2M-1}), \\ \vec{a}(2) &=& (a_{2M}, a_{2M+1}, \cdots, a_{3M-1}), \\ M &=& N/3. \end{array}\right\} \tag{3.3}$$

The generalized Games-Chan algorithm for computing the LC of $\{a_i\}$, denote as LC($\{a_i\}$), is written as follows.

[Generalized Games-Chan algorithm over $GF(3)$]

(i) Initial values:
$N = 3^n$, $LC = 0$, $\vec{a} = Eq.(3.1)$.

(ii) Repeat the following 0)∼3) until $N = 1$.

0). From the given $\vec{a}$ in (3.2), compute
$\vec{b} = (b_0, b_1, \cdots, b_{M-1})$,
$\vec{c} = (c_0, c_1, \cdots, c_{M-1})$, $M = N/3$ by

$$\left.\begin{array}{rcl}\vec{b} &=& \vec{a}(0) + \vec{a}(1) + \vec{a}(2), \\ \vec{c} &=& 2\vec{a}(0) + \vec{a}(1).\end{array}\right\} \qquad (3.4)$$

1). If $\vec{b} = \vec{c} = \vec{0}$, then

$$\vec{a} \leftarrow \vec{a}(0), \quad N \leftarrow N/3$$

and go to 0).

2). If $\vec{b} = \vec{0}$ and $\vec{c} \neq \vec{0}$, then

$$\vec{a} \leftarrow \vec{c}, \quad LC \leftarrow LC + N/3, \quad N \leftarrow N/3$$

and go to 0).

3). If $\vec{b} \neq \vec{0}$, then

$$\vec{a} \leftarrow \vec{b}, \quad LC \leftarrow LC + 2N/3, \quad N \leftarrow N/3$$

and go to 0).

(iii) If $\vec{a} \neq \vec{0}$ with $N = 1$, then

$$LC \leftarrow LC + 1.$$

The final LC is equal to LC($\{a_i\}$).□

3.3 The proposed Algorithm for Computation the k-LC

The k-LC of a sequence $\{a_i\}$ over $GF(3)$ with period $N = 3^n$ is defined as

$$k\text{-LC}(\{a_i\}) = \min\{\text{LC}(\{a_i + e_i\}) | w_H(\vec{e}) \leq k\}, \qquad (3.5)$$

where $\{e_i\}$ is a sequence over $GF(3)$ with period N called the error sequence, $\vec{e} = (e_0, e_1, \cdots, e_{N-1})$ called the error vector and $w_H(\vec{e})$ is the Hamming weight of the vector $\vec{e}$. If I have no effective algorithm for computing the k-LC, I must repeatedly apply the generalized Games-Chan algorithm at the worst case

$$\sum_{i=0}^{k} 2^i \binom{N}{i} \tag{3.6}$$

times to the sequences $\{a_i + e_i\}$'s with different $\vec{e_i}$'s. However (3.6) is very large when either N or k is large.

In order to compute the k-LC of $\vec{a}$ in (3.1), I must check, in the step (ii) of the generalized Games-Chan algorithm, whether I can make first 1) $\vec{b} = \vec{c} = \vec{0}$ and next 2) $\vec{b} = \vec{0}$ if 1) is impossible under the condition that the minimum number of changes about the original $\vec{a}$ of length $N = 3^n$ necessary and sufficient for obtaining 1) or 2) with the additional condition that all the previous $\vec{b} = \vec{0}$ and $\vec{c} = \vec{0}$ are kept the same. This check can be made conveniently by introducing the following costs of $\vec{a}$, $\vec{b}$ and $\vec{c}$.

First the cost of $\vec{a} = (a_0, a_1, \cdots, a_{N-1})$, $N = 3^m$, is denoted as $3 \times N$ matrix $A = [A_{j,i}]$, ($j = 0, 1, 2$; $i = 0, 1, \cdots, N-1$), where $A_{j,i}$ is defined as the minimum number of changes about the original sequence with period 3^n necessary and sufficient for changing a_i to $a_i + j$ with the condition that all the previous $\vec{b} = \vec{0}$ and $\vec{c} = \vec{0}$ are kept the same.

Secondly the cost of $\vec{b} = (b_0, b_1, \cdots, b_{M-1})$, $M = N/3$, is denoted as $\vec{B} = (B_0, B_1, \cdots, B_{M-1})$, where B_i is defined as the minimum number of changes about the original sequences with period 3^n necessary and sufficient for changing b_i to 0 with the condition that all the previous $\vec{b} = \vec{0}$ and $\vec{c} = \vec{0}$ are kept the same.

Thirdly the cost of $\vec{c} = (c_0, c_1, \cdots, c_{M-1})$ is denoted as $\vec{C} = (C_0, C_1, \cdots, C_{M-1})$, where C_i is defied in the same way as B_i provided the condition $b_i = c_i = 0$ is substituted instead of the condition $b_i = 0$.

When $\vec{a} = (a_0, a_1, \cdots, a_{N-1})$ and its cost $A = [A_{j,i}]$ are given in step (ii), I can compute the cost $\vec{B} = (B_0, B_1, \cdots, B_{M-1})$, $M = N/3$, and the cost $\vec{C} = (C_0, C_1, \cdots, C_{M-1})$ in the following way.

The cost B_i is equal to the minimum value of $A_{d_0,i} + A_{d_1,i+M} + A_{d_2,i+2M}$ under the condition that $d_0 + d_1 + d_2 = 2b_i$, or equivalently

$d_0 = s$, $d_1 = t$, $d_2 = 2(s+t+b_i)$ with $s, t \in \{0,1,2\}$, since $b_i = a_i + a_{i+M} + a_{i+2M}$ and I try to make $(a_i + d_0) + (a_{i+M} + d_1) + (a_{i+2M} + d_2) = 0$. Therefore I have

$$B_i = \min_{s,t\in\{0,1,2\}} [A_{s,i} + A_{t,i+M} + A_{2(s+t+b_i),i+2M}]. \tag{3.7}$$

As to the cost $\vec{C} = (C_0, C_1, \cdots, C_{M-1})$, I try to make $(a_i + d_0) + (a_{i+M} + d_1) + (a_{i+2M} + d_2) = 0$ and $2(a_i + d_0) + (a_{i+M} + d_1) = 0$, which gives two conditions of $d_0 + d_1 + d_2 = 2b_i$ and $2d_0 + d_1 = 2c_i$ and three solutions of $d_0 = s$, $d_1 = s + 2c_i$ and $d_2 = s + 2b_i + c_i$ with $s \in \{0,1,2\}$. Therefore I have

$$C_i = \min_{s\in\{0,1,2\}} [A_{s,i} + A_{s+2c_i,i+M} + A_{s+2b_i+c_i,i+2M}]. \tag{3.8}$$

It is obvious from (3.7) and (3.8) that $B_i \le C_i$.

The decision of 1), 2) and 3) in step (ii) is made by

$$\left.\begin{aligned} TB &= B_0 + B_1 + \cdots + B_{M-1}, \\ TC &= C_0 + C_1 + \cdots + C_{M-1} \end{aligned}\right\} \tag{3.9}$$

in the following way, i.e.,

$$\left.\begin{aligned} &1). \iff TC \le k, \\ &2). \iff TB \le k < TC, \\ &3). \iff k < TB. \end{aligned}\right\} \tag{3.10}$$

Let $\vec{a}' = (a'_0, a'_1, \cdots, a'_{M-1})$, $M = N/3$, be obtained from $\vec{a} = (a_0, a_1, \cdots, a_{N-1})$ by choosing one of 1), 2) or 3) in step (ii). The computation of the cost $A' = [A'_{j,i}]$ of $\vec{a}'$ can be made in the following way.

First consider the case of 1), to find $A'_{j,i}$ I try to make $a'_i = a_i$ to $a'_i + j$ by substituting $a_i + d_0$, $a_{i+M} + d_1$ and $a_{i+2M} + d_2$ into a_i, a_{i+M} and a_{i+2M}, respectively, under the condition that I keep $(a_i + d_0) + (a_{i+M} + d_1) + (a_{i+2M} + d_2) = 0$ and $2(a_i + d_0) + (a_{i+M} + d_1) = 0$. These conditions on d_0, d_1, d_2 become as $d_0 = j$, $d_0 + d_1 + d_2 = 2b_i$ and $2d_0 + d_1 = 2c_i$, or equivalently $d_0 = j$, $d_1 = j + 2c_i$ and $d_2 = j + 2b_i + c_i$. Therefore I have

$$A'_{j,i} = A_{j,i} + A_{j+2c_i,i+M} + A_{j+2b_i+c_i,i+2M} \tag{3.11}$$

in case of 1).

Second consider the case of 2), to find $A'_{j,i}$ I try to make $a'_i = c_i = 2a_i + a_{i+M}$ to $a'_i + j$ by substituting $a_i + d_0$, $a_{i+M} + d_1$ and $a_{i+2M} + d_2$ into a_i, a_{i+M} and a_{i+2M}, respectively, under the condition that $(a_i + d_0) + (a_{i+M} + d_1) + (a_{i+2M} + d_2) = 0$. The condition on d_0, d_1 and d_2 becomes $2d_0 + d_1 = j$ and $d_0 + d_1 + d_2 = 2b_i$, or equivalently $d_0 = s \in \{0, 1, 2\}$, $d_1 = s + j$ and $d_2 = s + 2(b_i + j)$. Therefore I have

$$A'_{j,i} = \min_{s \in \{0,1,2\}} \{A_{s,i} + A_{s+j,i+M} + A_{s+2(b_i+j),I+2M}\} \tag{3.12}$$

in case of 2).

Third consider the case of 3), to find $A'_{i,j}$ I try to make $a'_i = b_i = a_i + a_{i+M} + a_{i+2M}$ to $a'_i + j$ by substituting $a_i + d_0$, $a_{i+M} + d_1$ and $a_{i+2M} + d_2$ into a_i, a_{i+M} and a_{i+2M}, respectively. The 3-tuple (d_0, d_1, d_2) must satisfy $d_0 + d_1 + d_2 = j$ in this case and I have

$$A'_{j,i} = \min_{s,t \in \{0,1,2\}} \{A_{s,i} + A_{t,i+M} + A_{j+2(s+t),i+2M}\} \tag{3.13}$$

in case of 3).

Above preparations give the following algorithm for computing the k-LC of $\{a_i\}$ over $GF(3)$ with period $N = 3^n$.

[**Algorithm for computing the k-LC**]

(i) Initial values:
$N = 3^n$, $k\text{-}LC = 0$, $\vec{a} = Eq.(1)$,

$$A = \underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \end{bmatrix}}_{N}.$$

(ii) Repeat the following 0)$\sim$3) until $N = 1$.

0). From the given $\vec{a}$ in (2), compute $\vec{b}$ and $\vec{c}$ by (4).
Compute costs $\vec{B} = (B_0, B_1, \cdots, B_{M-1})$ and $\vec{C} = (C_0, C_1, \cdots, C_{M-1})$ by (3.7) and (3.8), respectively.
Compute costs TB and TC by (3.9).

1). If $TC \leq k$, then

$$\vec{a} \leftarrow \vec{a}(0), \;\; A \leftarrow A', \;\; N \leftarrow N/3,$$

where $A' = [A'_{j,i}]$ is computed by (3.11), and go to 0).

2). If $TB \leq k < TC$, then

$$\begin{aligned} &\vec{a} \leftarrow 2\vec{a}(0) + \vec{a}(1), \\ &k\text{-}LC \leftarrow k\text{-}LC + N/3, \\ &A \leftarrow A', \;\; N \leftarrow N/3, \end{aligned}$$

where $A' = [A'_{j,i}]$ is computed by (3.12), and go to 0).

3). If $k < TB$, then

$$\begin{aligned} &\vec{a} \leftarrow \vec{a}(0) + \vec{a}(1) + \vec{a}(2), \\ &k\text{-}LC \leftarrow k\text{-}LC + 2N/3, \\ &A \leftarrow A', \;\; N \leftarrow N/3, \end{aligned}$$

where $A' = [A'_{j,i}]$ is computed by (3.13), and go to 0).

(iii) If $A_{2a_0,0} > k$ for $N = 1$, then

$$k\text{-}LC \leftarrow k\text{-}LC + 1.$$

The final k-LC is equal to the k-LC($\{a_i\}$). □

I define

$$MAS(u) = \sum_{i=0}^{N-1} \min\{A_{0,i}, A_{1,i}, A_{2,i}\} \tag{3.14}$$

concerning the matrix $A(u)$ at step u. I also write TB and TC at step u as $TB(u)$ and $TC(u)$ in order to make it clear that those are values at step u. Then I can prove the following proposition.

Proposition 3.1: (a) $MAS(u) \leq TB(u) \leq TC(u)$.
(b) If $TC(u) \leq k$, then $MAS(u+1) = TC(u)$.
(c) If $TB(u) \leq k < TC(u)$, then $MAS(u+1) = TB(u)$.
(d) If $k < TB(u)$, then $MAS(u+1) = MAS(u)$.
(e) $0 = MAS(0) \leq MAS(1) \leq \cdots \leq MAS(n) \leq k$.
The proof of Proposition is rather easy, and so I will omit it.

The validity of this algorithm is clear, since (i) it uses the logic similar to that used in the Stamp-Martin algorithm [SM93], i.e., "apply the generalized Games-Chan algorithm, but if $\vec{c} \neq \vec{0}$ or $\vec{b} \neq \vec{0}$ and I can force $\vec{c} = \vec{b} = \vec{0}$ or $\vec{b} = \vec{0}$, I do so" and (ii) from (e) of Proposition 3.1, the minimum of $A_{0,0}, A_{1,0}, A_{2,0}$ is less than or equal to k.

3.4 Computation of an Error Vector

Computation of an error vector $\vec{e}$ appeared in (3.5) is straightforward, as shown below, if (i) I include additional $3 \times 3^{n-u}$ matrix $D(u)$ at step u in our algorithm for computing the k-LC and (ii) reading $D(u)$'s backward from $u = n$ to $u = 1$. Firstly the (j, i)-element of $D(u)$, denoted as $D(u)_{j,i}$ is the 3-tuple (d_0, d_1, d_2), $(d_0, d_1, d_2 \in \{0, 1, 2, \})$, satisfying $A'_{j,i} = A_{d_0,i} + A_{d_1,i+M}, A_{d_2,i+2M}$ in (3.11), (3.12) or (3.13), where $A_{j,i}$ is the value at step $(u-1)$ and $A'_{j,i}$ that at step u. I defined a vector $d(\vec{u}) = (\vec{d}(u)_0, \vec{d}(u)_1, \cdots, \vec{d}(u)_{M-1})$, where $M = 3^{n-u}$ and $\vec{d}(u)_i$ a 3-tuple over $GF(3)$. I have the following algorithm for computing an error vector.

[**Algorithm for computing an error vector**]

(i) Initial values:
Let $\vec{a} = (a_0)$ at step n and $u = n$.
$s = 2a_0$ if $A_{2a_0,0} \leq k$.
$s = \min\{A_{0,0}, A_{1,0}, A_{2,0}\}$ if $A_{2a_0,0} > k$.
Compute a 3-tuple $\vec{d}$ by $\vec{d}(n) = D(n)_{s,0}$.

(ii) Repeat the following calculation (a)-(b) until $u = 1$.

(a) Compute $\vec{d}(u-1) = (\vec{d}(u-1)_0, \vec{d}(u-1)_1, \cdots, \vec{d}(u-1)_{3M-1})$, $M = 3^{n-u}$ by

$$\left.\begin{array}{rcl} \vec{d}(u-1)_i & = & D(u-1)_{i,d_{0,i}}, \\ \vec{d}(u-1)_{i+M} & = & D(u-1)_{d_{i,1},i+M}, \\ \vec{d}(u-1)_{i+2M} & = & D(u-1)_{d_{i,2},i+2M}, \end{array}\right\}$$

where I write $\vec{d}(u)_i = (d_{0,i}, d_{1,i}, d_{2,i})$.

(b) $M \leftarrow 3M$, $u \leftarrow u - 1$.
If $u \neq 1$, then go to (a).

(iii) If $u = 1$ and $\vec{d}(1) = ((d_{0.0}, d_{1,0}, d_{2,0}),$ $(d_{0,1}, d_{1,1}, d_{2,1}), \cdots, (d_{0,M-1}, d_{1,M-1}, d_{2,M-1}))$, then I have an error vector by

$$e_i = d_{0,i}, \quad e_{i+M} = d_{1,i}, \quad e_{i+2M} = d_{2,i}.$$

The final $\vec{e}$ is desirable error vector. □

3.5 A Numerical Example

In this section I show the performance of the proposed algorithm and the profile of the k-LC about k.

Let $\{a_i\}$ be a sequence over $GF(3)$ with period $3^3 = 27$ whose one period is

$$\vec{a} = (020211010120111010220211010). \tag{3.15}$$

I will compute 3-LC ($k = 3$) and an error vector.

[Example 3.1]
(initial values)

$$\vec{a} = (020211010120111010220211010)$$

$$A = \begin{bmatrix} 000000000000000000000000000 \\ 111111111111111111111111111 \\ 111111111111111111111111111 \end{bmatrix}$$

(step 1)

$$\vec{b} = (000200000) \qquad \vec{B} = (000100000) \qquad TB = 1$$

$$\vec{c} = (100200000) \qquad \vec{C} = (200100000) \qquad TC = 3$$

$$\vec{a} = (020211010) \qquad k\text{-LC} = 0$$

$$A = \begin{bmatrix} 200100000 \\ 233333333 \\ 233233333 \end{bmatrix}$$

$$D(1) = \begin{bmatrix} (021)\underline{(000)(000)(010)(000)(000)(000)(000)(000)} \\ (102)\overline{(111)(111)(121)(111)(111)(111)(111)(111)} \\ \underline{(210)}(222)(222)(202)(222)(222)(222)(222)(222) \end{bmatrix}$$

(step 2)

$\vec{b} = (211) \qquad \vec{B} = (333) \qquad TB = 9$

$\vec{c} = (221) \qquad \vec{C} = (533) \qquad TC = 11$

$\vec{a} = (211) \qquad k\text{-LC} = 6$

$$A = \begin{bmatrix} 300 \\ 333 \\ 333 \end{bmatrix}$$

$$D(2) = \begin{bmatrix} (000)\underline{(000)(000)} \\ (100)(001)(001) \\ \underline{(200)}(002)(002) \end{bmatrix}$$

(step 3)

$\vec{b} = (1) \qquad \vec{B} = (3) \qquad TB = 3$

$\vec{c} = (2) \qquad \vec{C} = (3) \qquad TC = 3$

$\vec{a} = (2) \qquad k\text{-LC} = 6$

$$A = \begin{bmatrix} 9 \\ 9 \\ 3 \end{bmatrix}$$

$$D(3) = \begin{bmatrix} (011) \\ (122) \\ \underline{(200)} \end{bmatrix}$$

Finally k-LC$= 7 \quad (k = 3)$. Since $\vec{a} = (2)$ at step 3 and $A_{1,0} = 9 > k = 3$, so k-LC$= 6 + 1 = 7$.

[computing an error vector]

(initial value) $\quad s = 2$

(step 3) $\quad \vec{d}(3) = (200)$

(step 2) $\quad \vec{d}(2) = ((200)(000)(000))$

(step 1)

$\vec{d}(1) = ((210)(000)(000)(010)(000)(000)(000)(000)(000))$

The error vector is

$\vec{e} = (200000000100100000000000000)$.

Next there is table. 1 which shows the value of the k-LC about the allowed number of errors k.

[table. 1 profile of $\{a_i\}$ about k]

k	0	1	2	3	4	5	6	7	8
k-LC	27	15	15	7	7	7	7	7	7
k	9	10	11	12	13	14	15	16	17
k-LC	4	4	3	2	2	2	2	1	0

3.6 Conclusion

In this section I propose the algorithm for the k-LC of sequences over $GF(3)$ with period 3^n. In order to generalize into non-binary sequences, I try to extend the cost vector at the Stamp-Martin algorithm of sequences over $GF(2)$ with period 2^n to the cost matrix.

Furthermore I show a method for computing not only the k-LC but also the error vector of sequences over $GF(3)$ with period 3^n. In applications the error vector will play important roles. And an example of sequences over $GF(3)$ with period $3^3 = 27$ and the profile of the k-LC about k are shown.

Finally it is able to calculate the k-LC of general sequences over $GF(p^m)$ with period p^n.

Chapter 4

An Algorithm for the k-LC over $GF(p^m)$ with period p^n

4.1 Introduction

An algorithm for the k-error linear complexity of sequences over $GF(p^m)$ with period p^n is given, where p is a prime. The algorithm is derived by the generalized Games-Chan algorithm for the LC of sequences over $GF(p^m)$ with period p^n, and by using the modified cost not the same as that used in the Stamp-Martin algorithm for sequences over $GF(2)$ with period 2^n. A method is also given for computing an error vector which gives the k-LC.

The k-LC gives much more reasonable evaluation than the conventional LC for the randomness of a keystream in stream ciphers. It is desirable to find useful applications of the k-LC in the cryptanalysis of stream ciphers.

Unfortunately an effective algorithm for computing the k-LC has been known only for sequences over $GF(2)$ with period 2^n (the Stamp-Martin algorithm) [SM93]. The Stamp-Martin algorithm uses the Games-Chan algorithm [GC83] for computing the LC of sequences over $GF(2)$ with period 2^n. An alternative derivation of the Stamp-Martin algorithm [SM93] was recently given by the author et al. [KUI96-2]. This method[KUI96-2] can compute not only the k-LC but also an error vector which gives the k-LC.

Since the Games-Chan algorithm was generalized to the sequences over $GF(p^m)$ with period p^n, p a prime, by Ding, Xiao, and Shan [DXS91] and also by the author et al.[IM93], it seems to be possible to find a similar algorithm for the k-LC of sequences over $GF(p^m)$ with period p^n. Such a generalization was made by the author et al. [KUI96] in case of $p = 3$ and $m = 1$. In this chapter I give a complete description of the algorithm for the k-LC of sequences over $GF(p^m)$ with period p^n, p a prime. It is shown that both of the logic of our algorithm and its description are rather simple.

Firstly I show the generalized Games-Chan algorithm for the LC of sequences over $GF(p^m)$ with period p^n, where p is a prime and n and m are positive integers, in Section 4.2. This algorithm can be written very simply by the formula of vector $\vec{b}$ given in [UIK97].

In Section 4.3 the proposed algorithm for the k-LC of sequences over $GF(p^m)$ with period p^n is given after several preparations of the cost matrices of the vector $\vec{a}$ and vector $\vec{b}$. Moreover the property of the algorithm, named Proposition 4.1, is shown in order to prove the validity of the proposed algorithm.

Next in Section 4.4 I discuss about computing an error vector which gives the k-LC. This algorithm executes using the stored data for the value of changing at each step in order of backward when the k-LC is calculated similar to the algorithm of sequences over $GF(2)$ with period 2^n in Chapter 2 or over $GF(3)$ with period 3^n in Chapter 3, respectively.

Finally I show two numerical examples of computing the k-LC and an error vector. In Example 4.1 the performance of the proposed algorithm of a sequence over $GF(3)$ with period $3^3 = 27$ and $k = 3$ is given. In Example 4.2 the performance of the proposed algorithm of sequence over $GF(3^2)$ with period $3^3 = 27$ is given also.

On the other hand Blackburn [Bla94] gave an algorithm for the minimal polynomial of a periodic sequence with period N_0p^n, where p is a prime and $\gcd(N_0, p) = 1$, by jointly using the discrete Fourier transform for sequences with period N_0 and the Games-Chan algorithm for sequences with period p^n. In this dissertation, however, I will not try to find an algorithm for the k-LC of sequences with period N_0p^n, although Blackburn's algorithm looks very promising to generalize this algorithm to the most general case.

4.2 The Generalized Games-Chan Algorithm

In this chapter I will consider sequences over $GF(q)$ with period $N = p^n$, $n \geq 1$, where $q = p^m$ and p is a prime.

Let $\{a_i\} = \{a_0, a_1, a_2, \cdots\}$ be a sequence over $GF(q)$ with period $N = p^n$ and the first period of the sequence be denoted as

$$\vec{a}^{(N)} = (a_0^{(N)}, a_1^{(N)}, \cdots, a_{N-1}^{(N)}). \tag{4.1}$$

I will write $\vec{a}^{(N)}$ as

$$\vec{a}^{(N)} = (\vec{a}(0)^{(N)}, \cdots, \vec{a}(p-1)^{(N)}), \tag{4.2}$$

where

$$\vec{a}(j)^{(N)} = (a_{jM}^{(N)}, \cdots, a_{(j+1)M-1}^{(N)}), \quad M = N/p = p^{n-1}. \tag{4.3}$$

The LC of $\{a_i\}$ over $GF(q)$, denoted as $LC(\{a_i\})$, is defined as $LC(\{a_i\}) = L$ if

$$\begin{aligned} \sum_{i \geq 0} a_i x^{-(i+1)} &= \frac{a(x)^{(N)}}{x^N - 1} \\ &= \frac{g(x)}{(x-1)^L}, \end{aligned} \tag{4.4}$$

where $a(x)^{(N)} = a_0^{(N)} x^{N-1} + a_1^{(N)} x^{N-2} + \cdots + a_{N-1}^{(N)}$ and $g(1) \neq 0$ (Note that $x^{p^n} - 1 = (x-1)^{p^n}$). If $a(x)^{(N)}$ has a zero for the order $Z = z_0 + z_1 p + \cdots + z_{n-1} p^{n-1}$, $(0 \leq z_i \leq p-1)$, of zero at $x = 1$ in the order from z_{n-1} to z_0, where $z_i \in \{0, 1, \cdots, p-1\}$ for $i = 0, 1, \cdots, n-1$. Therefore I have $L = N - Z$. The Games-Chan algorithm determines z_{n-1}, z_{n-2}, $\cdots$, z_0 in the order by making repeated divisions of a polynomial with degree at most $p^{i+1} - 1$ by $(x-1)^{p^i} = x^{p^i} - 1$.

Let $M = N/p$ and $\vec{a}(j)^{(N)} = (a_{jM}^{(N)}, a_{jM+1}^{(N)}, \cdots, a_{(j+1)M-1}^{(N)})$ be an M-tuple for $j = 0, 1, \cdots, p-1$. I will rewrite $\vec{a}^{(N)} = (\vec{a}(0)^{(N)}, \vec{a}(1)^{(N)}, \cdots, \vec{a}(p-1)^{(N)})$. I show the generalized Games-Chan algorithm as follows[DXS91, IM93].

[**Generalized Games-Chan Algorithm**][DXS91, GC83, IM93]

(i) Initial values:
$N = pM = p^n, \quad LC = 0, \quad \vec{a}^{(N)} = $ Eq.(4.1).

(ii) Repeat the following 0)∼2) until $M = 1$.

0). From a given pM-tuple over $GF(q)$,

$$\vec{a}^{(pM)} = (\vec{a}(0)^{(pM)}, \cdots, \vec{a}(p-1)^{(pM)}), \tag{4.5}$$

compute the following M-tuples over $GF(q)$, $\vec{b}(u)^{(M)}$'s, $(u = 0, \cdots, p-2)$, by

$$\begin{aligned} \vec{b}(u)^{(M)} &= F_u(\vec{a}(0)^{(pM)}, \cdots, \vec{a}(p-1)^{(pM)}) \\ &= \sum_{j=0}^{p-u-1} c_{u,j}\vec{a}(j)^{(pM)}, \end{aligned} \tag{4.6}$$

where

$$F_u(x_0, \cdots, x_{p-1}) = \sum_{j=0}^{p-u-1} c_{u,j}x_j, \quad c_{u,j} = \binom{p-j-1}{u}. \tag{4.7}$$

1). Choose one of the following p cases.
Case 1:

$$\vec{b}(0)^{(M)} = \cdots = \vec{b}(p-2)^{(M)} = \vec{0}. \tag{4.8}$$

Case w, $(2 \leq w \leq p-1)$:

$$\vec{b}(0)^{(M)} = \cdots = \vec{b}(p-w-1)^{(M)} = \vec{0}, \quad \vec{b}(p-w)^{(M)} \neq \vec{0}. \tag{4.9}$$

Case p:

$$\vec{b}(0)^{(M)} \neq \vec{0}. \tag{4.10}$$

2). If **Case** w, $(1 \leq w \leq p)$ is chosen, then

$$\vec{a}^{(M)} \leftarrow F_{p-w}(\vec{a}(0)^{(pM)}, \cdots, \vec{a}(p-1)^{(pM)})$$

$$LC \leftarrow LC + (w-1)M, \qquad M \leftarrow M/p$$

and go to 0). Here I use $F_{p-1}(x_0, \cdots, x_{p-1}) = x_0$.

(iii). Let $\vec{a}^{(1)} = (a_0^{(1)})$. If $a_0^{(1)} \neq 0$, then

$$LC \leftarrow LC + 1.$$

The final LC is equal to $\mathrm{LC}(\{a_i\})$.□

Note that the Case w $(1 \leq w \leq p)$ corresponds to the case where $(x^M - 1)^{p-w}$ divides $a(x)^{(pM)}$ but $(x^M - 1)^{p-w+1}$ does not. The formula (4.6) and (4.7) are given in [UIK97]. Note also that for $\vec{a}^{(pM)} \neq \vec{0}$ only one of the p cases, i.e., Case 1, $\cdots$, Case p, occurs [UIK97].

4.3 An Algorithm for Computing the k-LC

The k-LC of a sequence $\{a_i\}$ over $GF(q)$ with period $N = p^n$ is defined as

$$k\text{-LC}(\{a_i\}) = \min\{\mathrm{LC}(\{a_i + e_i\}) | w_H(\vec{e}) \leq k\}, \tag{4.11}$$

where $\{e_i\}$ is an error sequence over $GF(q)$ with period N and $w_H(\vec{e})$ is the Hamming weight of the first N-tuple, $\vec{e} = (e_0, e_1, \cdots, e_{N-1})$, of $\{e_i\}$, i.e., the number of nonzero e_j 's. I will call $\vec{e}$ an error vector. If I have no effective algorithm for computing the k-LC, I must repeatedly apply the Games-Chan algorithm at the worst case

$$\sum_{i=0}^{k} (q-1)^i \binom{N}{i} \tag{4.12}$$

times to the sequences $\{a_i + e_i\}$'s with all the possible $\vec{e}$'s having Hamming weight $\leq k$. However (4.12) becomes very large even for moderate N and k.

In order to compute the k-LC of $\vec{a}$ in (4.1), I must try to force Case w to happen for as small w as possible in step (ii) of the Games-Chan algorithm under the condition that the minimum number of changes in the original $\vec{a}^{(N)}$ necessary and sufficient for forcing Case w to happen is less than or equal to k. This logic is the same as that used in the Stamp-Martin algorithm [SM93]. This can be done conveniently by introducing the following cost of $\vec{a}^{(M)}$ and the costs of $\vec{b}(u)^{(M)}$'s, $(0 \leq u \leq p-2)$.

In the following I will write

$$GF(q) = \{\alpha_0 = 0,\ \alpha_1, \cdots,\ \alpha_{q-1}\}, \tag{4.13}$$

and

$$\vec{b}(u)^{(M)} = (b_{u,0}^{(M)}, \cdots, b_{u,M-1}^{(M)}). \tag{4.14}$$

Firstly the cost of $\vec{a}^{(M)}$ is denoted as a $q \times M$ matrix

$$AC(M) = [A(h,i)_M], \quad (0 \le h \le q-1; 0 \le i \le M-1), \tag{4.15}$$

where $A(h,i)_M$ is the minimum number of changes in $\vec{a}^{(N)}$ necessary and sufficient for changing $a_i^{(M)}$ to $a_i^{(M)} + \alpha_h$ under the condition that forcing Case w to happen is not altered. In the following discussions I will often use the notation $A(\alpha_h, i)_M$ for $A(h,i)_M$.

Secondly the costs of $\vec{b}(u)^{(M)}$'s, $(0 \le u \le p-2)$, are denoted as a $(p-1) \times M$ matrix

$$BC(M) = [B(u,i)_M], \quad (0 \le u \le p-2; 0 \le i \le M-1), \tag{4.16}$$

where $B(u,i)_M$ is the minimum number of changes in $\vec{a}^{(N)}$ necessary and sufficient for making $b_{0,i}^{(M)} = \cdots = b_{u,i}^{(M)} = 0$. I will define the total cost of $\vec{b}(u)^{(M)}$ as

$$TB(u)_M = \sum_{i=0}^{M-1} B(u,i)_M, \quad (0 \le u \le p-2), \tag{4.17}$$

which means the minimum number of changes in $\vec{a}^{(N)}$ necessary and sufficient for making $\vec{b}(0)^{(M)} = \cdots = \vec{b}(u)^{(M)} = \vec{0}$.

In step (ii) of the Games-Chan algorithm I can force Case 1 to happen if $TB(p-2)_M \le k$, Case w, $(2 \le w \le p-1)$ to happen if $TB(p-w-1)_M \le k < TB(p-w)_M$, and Case p to happen if $k < TB(0)_M$, respectively.

The following initial value, $AC(N) = [A(h,i)_N]$, $(N = p^n;\ 0 \le h \le q-1;\ 0 \le i \le N-1\ )$, is obvious from the definition.

$$A(h,i)_N = \begin{cases} 0, & \text{if } h = 0, \\ 1, & \text{if } h \ne 0. \end{cases} \tag{4.18}$$

I can compute $BC(M)$ and $AC(M)$ from $AC(pM)$ in the following way for $M = p^{n-1}, \cdots, p^0$ in the order.

$BC(M) = [B(u,i)_M]$ can be computed from $AC(pM) = [A(h,i)_{pM}]$ by

$$B(u,i)_M = \min\left\{ \sum_{j=0}^{p-1} A(e_j, i+jM)_{pM} \middle| \vec{e} \in D(u,i)_M \right\}, \qquad (4.19)$$

where

$$\vec{e} = (e_0, \cdots, e_{p-1}) \in [GF(q)]^p,$$

and

$$D(u,i)_M = \{\vec{e} \mid F_j(e_0,\cdots,e_{p-1}) + b_{j,i}^{(M)} = 0, \quad (0 \le j \le u)\}. \quad (4.20)$$

Note that $D(u,i)_M$ is the set of all the $\vec{e}$'s which can make $b_{0,i}^{(M)} = \cdots = b_{u,i}^{(M)} = 0$.

The computation of $AC(M)$ from $AC(pM)$ depends on the case chosen at step (ii). If I can force Case w, $(1 \le w \le p)$, to happen, I must choose $\vec{e} = (e_0, ..., e_{p-1})$ such that

$$\begin{array}{rlll} & F_{p-w}(a_i^{(pM)} + e_0, \cdots, a_{i+(p-1)M}^{(pM)} + e_{p-1}) & & \\ = & F_{p-w}(a_i^{(pM)}, \cdots, a_{i+(p-1)M}^{(pM)}) + F_{p-w}(e_0,\cdots,e_{p-1}) & & \\ = & a_i^{(M)} \quad + & F_{p-w}(e_0,\cdots,e_{p-1}) & \\ = & a_i^{(M)} \quad + & \alpha_h & \end{array} \qquad (4.21)$$

and

$$F_j(a_i^{(pM)} + e_0, \cdots, a_{i+(p-1)M}^{(pM)} + e_{p-1}) = b_{j,i}^{(M)} + F_j(e_0,\cdots,e_{p-1}) = 0 \qquad (4.22)$$

for $0 \le j \le p-w-1$. Therefore I have

$$A(h,i)_M = \min\left\{ \sum_{j=0}^{p-1} A(e_j, i+jM)_{pM} \middle| \vec{e} \in \hat{D}(h,i)_M^w \right\}, \qquad (4.23)$$

where

$$\hat{D}(h,i)_M^1 = \left\{ \vec{e} \,\middle|\, \begin{array}{l} F_j(e_0,\cdots,e_{p-1}) + b_{j,i}^{(M)} = 0, \quad (0 \le j \le p-2), \\ e_0 - \alpha_h = 0, \end{array} \right\} \qquad (4.24)$$

$$\hat{D}(h,i)^w_M = \left\{ \vec{e} \;\middle|\; \begin{array}{ll} F_j(e_0,\cdots,e_{p-1}) + b^{(M)}_{j,i} = 0, & (0 \le j \le p-w-1), \\ F_{p-w}(e_0,\cdots,e_{p-1}) - \alpha_h = 0, & \end{array} \right\} \tag{4.25}$$

for $2 \le w \le p-1$, and

$$\hat{D}(h,i)^p_M = \{\vec{e} \mid F_0(e_0,\cdots,e_{p-1}) - \alpha_h = 0,\}, \tag{4.26}$$

respectively.

I will keep a record of the following p-tuple

$$\vec{e}(h,i)_M = (e_0,\cdots,e_{p-1}) \in [GF(q)]^p \tag{4.27}$$

found at the computation of $A(h,i)_M$ in (4.23) in such a way as $\vec{e} = \vec{e}(h,i)_M$ gives $A(h,i)_M$ in the right-hand side of (4.23). In general $\vec{e}(h,i)_M$ is not unique. The record of $\vec{e}(h,i)_M$'s is a $q \times M$ matrix with p-tuple elements

$$E(M) = [\vec{e}(h,i)_M] \tag{4.28}$$

and will be used for computing an error vector which gives the k-LC.

The proposed algorithm for computing the k-LC of $\{a_i\}$ in (4.1) is written as follows.

[Algorithm for Computing the k-LC]

(i) Initial values:
$N = pM = p^n$, k-LC=0, $\vec{a}^{(N)} = Eq.(4.1)$,
$AC(N) = [A(h.i)_N] = Eq.(4.18)$

(ii) Repeat the following 0)∼2) until $M = 1$.

0). Compute $BC(M)$ by (4.19)-(4.20) and $TB(0)_M,\cdots,TB(p-2)_M$ by (4.17).

1). Choose
Case 1 if $TB(p-2)_M \le k$,
Case w, $(2 \le w \le p-1)$, if $TB(p-w-1)_M \le k < TB(p-w)_M$,
and Case p if $k < TB(0)_M$, respectively.

2). If Case w, $(1 \le w \le p)$, is chosen, then

$$\vec{a} \leftarrow F_{p-w}(\vec{a}(0)^{(pM)},\cdots,\vec{a}(p-1)^{(pM)}),$$

$$k\text{-LC} \leftarrow k\text{-LC} + (w-1)M.$$

Compute $AC(M)$ by (4.23)-(4.26) and $E(M)$ by (4.27) and (4.28).
If $M \neq 1$, then

$$M \leftarrow M/p$$

and go to 0).

(iii) Let $\vec{a}^{(1)} = (a_0^{(1)})$ and $AC(1) = [A(h,0)_1]$. If $A(-a_0^{(1)},0)_1 > k$, then

$$k\text{-LC} \leftarrow k\text{-LC} + 1.$$

The final *k-LC* is equal to the k-LC($\{a_i\}$). □

The validity of this algorithm can be shown by using the following Proposition 4.1.

Proposition 4.1: Let

$$MTAC(M) = \sum_{i=0}^{M-1} \min\{A(h,i)_M \mid 0 \leq h \leq q-1\}. \qquad (4.29)$$

(a) $MTAC(pM) \leq TB(0)_M \leq \cdots \leq TB(p-2)_M$.

(b) If $TB(p-2)_M \leq k$, then $MTAC(M) = TB(p-2)_M$.

(c) If $TB(p-w-1)_M \leq k < TB(p-w)_M$, then $MTAC(M) = TB(p-w-1)_M$.

(d) If $k < TB(0)_M$, then $MTAC(M) = MTAC(pM)$.

(e) $0 = MTAC(p^n) \leq MTAC(p^{n-1}) \leq \cdots \leq MTAC(p^0) \leq k$.

□

Proof:
(a) $TB(u)_M \leq TB(u+1)_M$ is obvious from (4.17) and (4.19) and (4.20).
$MTAC(pM) \leq TB(0)_M$ is also obvious, since

$$MTAC(pM) = \sum_{i=0}^{M-1}\sum_{j=0}^{p-1} \min\{A(h, i+jM)_{pM} \mid 0 \leq h \leq q-1\}$$

and

$$TB(0)_M = \sum_{i=0}^{M-1} \min\left\{ \sum_{j=0}^{p-1} A(e_j, i+jM)_{pM} \,\middle|\, F_0(e_0,\cdots,e_{p-1}) + b_{0,i}^{(M)} = 0 \right\}.$$

(b) If $TB(p-2)_M \leq k$, then I can force Case 1 to happen in step (ii). In this case $AC(M)$ is given by (4.21) and (4.22). I have $MTAC(M) = TB(p-2)_M$, since

$$\bigcup_{0\leq h\leq q-1} \hat{D}(h,i)_M^1 = D(p-2,i)_M$$

from (4.20) and (4.22).
(c) In this case I can force Case w to happen in step (ii) and $AC(M)$ is given by (4.23) and (4.25). I have $MTAC(M) = TB(p-w-1)_M$ in the same reason as (b).
(d) The proof is the same as (b) and (c).
(e) From (a)-(d) I have $MTAC(pM) \leq MTAC(M)$. $MTAC(p^n) = 0$ is obvious from (18). From (b)-(d) have $MTAC(M) \leq k$ if $MTAC(pM) \leq k$.

Consider the cost matrix $AC(1) = [A(h,0)_1]$ of $\vec{a}^{(1)} = (a_0^{(1)})$. Let

$$A(s,0)_1 = \min\{A(h,0)_1 \mid 0 \leq h \leq q-1\}. \tag{4.30}$$

I have $A(s,0)_1 \leq k$ from (e) of Proposition 1. This means that I can change $a_0^{(1)}$ to $a_0^{(1)} + \alpha_s$ under the condition that the minimum number of changes in the original $\vec{a}^{(N)}$ necessary and sufficient for making this change is less than or equal to k. Therefore the validity of this algorithm is shown.

4.4 Computation of an Error Vector

After performing the proposed algorithm for computing the k-LC of $\{a_i\}$, finding an error vector

$$\vec{e} = (e_0,\cdots,e_{N-1}) \tag{4.31}$$

which gives the k-LC is straight by tracing the step (ii) in the reverse order, i.e., from $M = p^0$ to $M = p^{n-1}$.

[**Computing an Error Vector**]

(i) Initial values:
$M = 1, \quad \vec{a}^{(1)} = (a_0^{(1)}), \quad AC(1) = [A(h,0)_1],$
$A(s,0)_1 = Eq.(4.30).$
If $A(-a_0^{(1)},0)_1 \leq k$, then $\vec{e}(1) = \vec{e}(-a_0^{(1)},0)_1$.
If $A(-a_0^{(1)},0)_1 > k$, then $\vec{e}(1) = \vec{e}(s,0)_1$.

(ii) Repeat the following computation of

$$\vec{e}\,'(pM) = (\vec{e}\,'_0, \cdots, \vec{e}\,'_{pM-1})$$

from $\vec{e}(M) = (\vec{e}_0, \cdots, \vec{e}_{M-1})$ by using $E(pM)$ until $pM = p^{n-1}$.
If $\vec{e}_i = (e_{i,0}, \cdots, e_{i,p-1})$, then

$$\vec{e}\,'_{i+jM} = \vec{e}(e_{i,j}, i+jM)_{pM}, \quad (0 \leq j \leq p-1; 0 \leq i \leq M-1).$$

(iii) Let $M = p^{n-1}$ and $\vec{e}(M) = (\vec{e}_0, \cdots, \vec{e}_{M-1})$. An error vector $\vec{e}$ in (29) can be computed as

$$e_{i+jM} = e_{i,j}, \quad (0 \leq i \leq M-1; 0 \leq j \leq p-1)$$

if $\vec{e}_i = (e_{i,0}, \cdots, e_{i,p-1})$.

□

4.5 Numerical Examples

[**Example 4.1**]
Let $\{a_i\}$ be a sequence over $GF(3) = \{\alpha_0 = 0, \ \alpha_1 = 1, \ \alpha_2 = 2\}$ with period $3^3 = 27$ whose one period is

$$\vec{a} = (020211010120111010220211010). \qquad (4.32)$$

I will compute the 3-LC ($k = 3$) and an error vector. In order to compute an error vector $\vec{e}$, I will compute the matrices $E(9), E(3), E(1)$ at each step in this example.

[Computing the k-LC]

(initial values) $\vec{a}^{(27)} = (020211010120111010220211010)$

$$AC(27) = \begin{bmatrix} 000000000000000000000000000 \\ 111111111111111111111111111 \\ 111111111111111111111111111 \end{bmatrix}$$

(step 1; M=9) $\vec{b}(0)^{(9)} = (000200000) \qquad \vec{b}(1)^{(9)} = (100200000)$

$$BC(9) = \begin{bmatrix} 000100000 \\ 200100000 \end{bmatrix} \quad \begin{matrix} TB(0)_9 = 1 \\ TB(1)_9 = 3 \end{matrix}$$

$\vec{a}^{(9)} = (020211010) \qquad k\text{-LC} = 0$

$$AC(9) = \begin{bmatrix} 200100000 \\ 233333333 \\ 233233333 \end{bmatrix}$$

$$E(9) = \begin{bmatrix} (021)(000)(000)(010)(000)(000)(000)(000)(000) \\ (102)\overline{(111)(111)(121)(111)(111)(111)(111)(111)} \\ \underline{(210)}(222)(222)(202)(222)(222)(222)(222)(222) \end{bmatrix}$$

(step 2; M=3) $\vec{b}(0)^{(3)} = (211) \qquad \vec{b}(1)^{(3)} = (221) \quad BC(9) = \begin{bmatrix} 333 \\ 533 \end{bmatrix}$

$$\begin{matrix} TB(0)_3 = 9 \\ TB(1)_3 = 11 \end{matrix} \quad \vec{a}^{(3)} = (211) \qquad k\text{-LC} = 6$$

$$AC(3) = \begin{bmatrix} 300 \\ 333 \\ 333 \end{bmatrix} \quad E(3) = \begin{bmatrix} (000)(000)(000) \\ (100)\overline{(001)(001)} \\ \underline{(200)}(002)(002) \end{bmatrix}$$

(step 3; M=1) $\vec{b}(0)^{(1)} = (1) \qquad \vec{b}(1)^{(1)} = (2) \quad BC(1) = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$

$TB(0)_1 = 3$
$TB(1)_1 = 3$

$$\vec{a}^{(1)} = (2) \qquad k\text{-LC} = 6 \quad AC(1) = \begin{bmatrix} 9 \\ 9 \\ 3 \end{bmatrix} \quad E(1) = \begin{bmatrix} (011) \\ (122) \\ \underline{(200)} \end{bmatrix}$$

Since $\vec{a}^{(1)} = (2)$ and $A(-2,0)_1 = A(1,0)_1 = 9 > k = 3$, therefore k-LC$= 6 + 1 = 7$. Finally k-LC$= 7 \quad (k = 3)$.□

[Computation of an error vector]

(initial value) $s = 2$

(step 3; M=1) $\vec{e}(1) = (200)$

(step 2; M=3) $\vec{e}(3) = ((200)(000)(000))$

(step 1; M=9) $\vec{e}(9) = ((210)(000)(000)(010)(000)(000)$
$(000)(000)(000))$

The error vector is $\vec{e} = (2000000001001000000000000000)$.□

[Example 4.2]

Let $\{a_i\}$ be a sequence over $GF(3^2)$ with period $3^3 = 27$. Let β be a primitive element of $GF(3^2) = \{\alpha_0 = \beta^* = 0,\ \alpha_1 = \beta^0 = 1,\ \alpha_2 = \beta^1,\ \cdots,\ \alpha_8 = \beta^7\}$ defined as $\beta^2 = 2\beta + 1$. In the following computations I will denote $\beta^*,\ \beta^i$ as $*,\ i$, respectively, for simplicity.

Let us write the first period of $\{a_i\}$ as

$$\vec{a} = (012345670012345671012345672). \tag{4.33}$$

I will compute the 3-LC ($k = 3$) and an error vector. In order to compute an error vector $\vec{e}$, I will compute the matrices $E(9), E(3), E(1)$ at each step in this example.

[Computation of the k-LC]

(initial values) $\vec{a}^{(27)} = (012345670012345671012345672)$

$$AC(27) = \begin{bmatrix} 000000000000000000000000000 \\ 111111111111111111111111111 \\ 111111111111111111111111111 \\ 111111111111111111111111111 \\ 111111111111111111111111111 \\ 111111111111111111111111111 \\ 111111111111111111111111111 \\ 111111111111111111111111111 \\ 111111111111111111111111111 \end{bmatrix}$$

(step 1; M=9) $\vec{b}(0)^{(9)} = (********4)$ $\quad \vec{b}(1)^{(9)} = (********6)$

$$BC(9) = \begin{bmatrix} 000000001 \\ 000000002 \end{bmatrix} \quad \begin{matrix} TB(0)_9 = 1 \\ TB(1)_9 = 2 \end{matrix}$$

$\vec{a}^{(9)} = (012345670) \quad k\text{-LC} = 0$

$$AC(9) = \begin{bmatrix} 000000002 \\ 333333333 \\ 333333333 \\ 333333333 \\ 333333333 \\ 333333333 \\ 333333332 \\ 333333332 \\ 333333333 \end{bmatrix}$$

$$E(9) = \begin{bmatrix} \underline{(***)} & \underline{(***)} & \underline{(***)} & \underline{(***)} & \underline{(***)} & \underline{(***)} & \underline{(***)} & \underline{(***)} & (*21) \\ (000) & (000) & (000) & (000) & (000) & (000) & (000) & (000) & (037) \\ (111) & (111) & (111) & (111) & (111) & (111) & (111) & (111) & (105) \\ (222) & (222) & (222) & (222) & (222) & (222) & (222) & (222) & (260) \\ (333) & (333) & (333) & (333) & (333) & (333) & (333) & (333) & (314) \\ (444) & (444) & (444) & (444) & (444) & (444) & (444) & (444) & \underline{(456)} \\ (555) & (555) & (555) & (555) & (555) & (555) & (555) & (555) & (57*) \\ (666) & (666) & (666) & (666) & (666) & (666) & (666) & (666) & (6*3) \\ (777) & (777) & (777) & (777) & (777) & (777) & (777) & (777) & (742) \end{bmatrix}$$

(step 2; M=3) $\vec{b}(0)^{(3)} = (456) \quad \vec{b}(1)^{(3)} = (234) \quad BC(9) = \begin{bmatrix} 333 \\ 666 \end{bmatrix}$

$\begin{matrix} TB(0)_3 = 9 \\ TB(1)_3 = 18 \end{matrix} \quad \vec{a}^{(3)} = (456) \quad k\text{-LC} = 6$

$$AC(3) = \begin{bmatrix} 002 \\ 333 \\ 333 \\ 333 \\ 333 \\ 333 \\ 332 \\ 332 \\ 333 \end{bmatrix} \quad E(3) = \begin{bmatrix} \underline{(***)(***)}(***) \\ (**0)(**0)(**0) \\ (**1)(**1)(**1) \\ (**2)(**2)(**2) \\ (**3)(**3)(**3) \\ (**4)(**4)\underline{(**4)} \\ (**5)(**5)(**5) \\ (**6)(**6)(**6) \\ (**7)(**7)(**7) \end{bmatrix}$$

(step 3; M=1) $\vec{b}(0)^{(1)} = (0) \quad \vec{b}(1)^{(1)} = (3) \quad BC(1) = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$

$TB(0)_1 = 3$
$TB(1)_1 = 5$

$$\vec{a}^{(1)} = (3) \qquad k\text{-LC} = 7 \quad AC(1) = \begin{bmatrix} 3 \\ 6 \\ 6 \\ 6 \\ 6 \\ 5 \\ 5 \\ 5 \\ 6 \end{bmatrix} \quad E(1) = \begin{bmatrix} \underline{(**4)} \\ (*00) \\ (*13) \\ (*27) \\ (*31) \\ (*4*) \\ (*56) \\ (*65) \\ (*72) \end{bmatrix}$$

Since $\vec{a}^{(1)} = (3) = (\beta^3)$ and $A(-\beta^3, 0)_1 = A(\beta^7, 0)_1 = A(8, 0)_1 = 6 > k = 3$, therefore k-LC$= 7 + 1 = 8$. Finally k-LC$= 8 \quad (k = 3)$.
□

[Computation of an error vector]

(initial value) $s = 2$

(step 3; M=1) $\vec{e}(1) = (**4)$

(step 2; M=3) $\vec{e}(3) = ((***)(***)(**4))$

(step 1; M=9) $\vec{e}(9) = ((***)(***)(***)(***)(***)(***)(***)(***)(456))$

The error vector is $\vec{e} = (********4********5********6)$.□

Although necessary computations are simple additions and comparisons, the computational complexity increases rapidly in case of the large w or equivalently the large k-LC because of the fact that from (4.25) the number of the possible $\vec{e}$'s in $\hat{D}(h, i)^w_M$ is equal to q^{w-1}.

The amount of the memories necessary for performing the computation is the same independent of the value of the k-LC.

4.6 Conclusion

Firstly the Stamp-Martin algorithm [SM93] for computing the k-LC of sequences over $GF(2)$ with period 2^n is generalized into sequences over $GF(p^m)$ with period p^n, where p is a prime.

Secondly the proposed algorithm can compute not only the k-LC but also an error vector which gives the k-LC. Computation of an error vector is important in applications of the k-LC.

The generalized Games-Chan algorithm used in the proposed algorithm needs memories for the entire period of the given sequence in computation, which may cause the disadvantage compared with the Berlekamp-Massey algorithm when the period is very large and the k-LC is small. However as to the computation of the k-LC there seems to be very difficult to find an effective algorithm by using the Berlekamp-Massey algorithm.

Further generalization of the algorithm for the most general sequences with period N_0p^n, $\gcd(N_0, p) = 1$, by using Blackburn's algorithm [Bla94] is interesting.

Chapter 5

Another Algorithm for the k-LC over $GF(p^m)$ with period p^n Using the Shift and Offset

5.1 Introduction

In this chapter another algorithm is given for the k-LC of sequences over $GF(p^m)$ with period p^n, where p is a prime. The algorithm is different from the previous one given by the author at Chapter 4 in the following two points and can be said to be a generalization of the Stamp-Martin algorithm for the k-LC of binary sequences with period 2^n. (1) The value of k decreases as the proceeding of computations. (2) The error vector can be obtained at the same time when the k-LC is obtained. The key ideas of the algorithm are the "shift" and the "offset" of the cost matrix, which are introduced by the authors to derive the Stamp-Martin algorithm for binary sequences from our previous algorithm.

In Chapter 4 the author[KUI96, KUI96-2, KUI99] gave an algorithm for the k-LC of sequences over $GF(p^m)$ with period p^n, where p is a prime. The algorithm can find not only the k-LC but also an error vector which gives the k-LC. The algorithm uses the generalized version of the Games-Chan algorithm for computing the LC of sequences over $GF(p^m)$ with period p^n [GC83, IM93, Bla94], and consists of n steps for sequences with period p^n. At an algorithm in Chapter 4[KUI96, KUI96-2, KUI99] the value of k remains the same throughout all the steps and the determination of an error vector can be performed only after finding the k-LC by reading the stored data in the reverse order from the last step to the first step.

In this chapter I will give another algorithm for the k-LC of sequences over $GF(p^m)$ with period p^n. This new algorithm can be said to be the generalization of the Stamp-Martin algorithm[SM93] for the binary sequences with period 2^n, since the value of k decreases as the proceeding of the steps. Another feature of this new algorithm is that an error vector can be obtained at the same time when the k-LC is obtained. The key ideas of this new algorithm are the "shift" and the "offset" of the cost matrix, which were introduced by the author[KUI96-2] to derive the Stamp-Martin algorithm from our previous algorithm in case of binary sequences.

In Section 5.2 I give several preparations for generalization of the Stamp-Martin algorithm into sequences over $GF(p^m)$ with period p^n and several conditions needed to generalize the Stamp-Martin algorithm. For instance the cost matrices of the vector $\vec{a}$ and the vector $\vec{b}$ are defined again using the shift and offset of the cost. For the generalized Games-Chan algorithm I need to choose only one case in p cases by computing the costs at each step.

The proposed algorithm in this chapter for the k-LC of sequences over $GF(p^m)$ with period p^n is given in Section 5.3. This algorithm decides the k-LC and an error vector at the same time.

Finally an example, denoted as Example 4.1, of the proposed algorithm for k-LC and an error vector of a sequence over $GF(3^2)$ with period $3^3 = 27$ is given in Section 5.4. From this example it seems for the proposed algorithm to be simpler than the algorithm in Chapter 4.

5.2 Generalization of the Stamp-Martin Algorithm

In this chapter I will use the same logic and cost matrices $AC(M)$ and $BC(M)$ as those in the previous algorithms[KUI96, KUI96-2, KUI99] except that the following two operations, i.e. the "shift" and the "offset" will be introduced in the computation of $AC(M)$.

Let $GF(q) = \{\alpha_0 = 0, \alpha_1, \cdots, \alpha_{q-1}\}$ and $\vec{b}(u)^{(M)} = (b(u)_0^{(M)}, b(u)_1^{(M)}, \cdots, b(u)_{M-1}^{(M)})$. Firstly the cost of $\vec{a}^{(M)}$ is denoted as a $q \times M$ matrix

$$AC(M) = [A(h,i)_M], 0 \leq h \leq q-1; 0 \leq i \leq M-1, \qquad (5.1)$$

where $A(h,i)_M$ is the minimum number of changes in the initial N-tuple $\vec{a}^{(N)}$ necessary and sufficient for changing $a_i^{(M)}$ into $a_i^{(M)} + \alpha_h$ under the condition that forcing **Case** w to happen is not altered. Secondly the cost of $\vec{b}(u)^{(M)}$ is denoted as a $(p-1) \times M$ matrix

$$BC(M) = [B(u,i)_M], 0 \leq u \leq p-2; 0 \leq i \leq M-1, \qquad (5.2)$$

where $B(u,i)_M$ is the minimum number of changes in the initial N-tuple $\vec{a}^{(N)}$ necessary and sufficient for making $b(0)_i^{(M)} = b(1)_i^{(M)} = \cdots = b(u)_i^{(M)} = 0$. I define the total cost of $\vec{b}(u)^{(M)}$'s for $u = 0, 1, \cdots, p-2$ as

$$TB(u)_M = \sum_{i=0}^{M-1} B(u,i)_M \qquad (5.3)$$

which means the minimum number of changes in the initial N-tuple $\vec{a}^{(N)}$ necessary and sufficient for making $\vec{b}(0)^{(M)} = \vec{b}(1)^{(M)} = \cdots = \vec{b}(u)^{(M)} = \vec{0}$. In step (ii) of the generalized Games-Chan algorithm I can force **Case** w to happen:

$$\begin{array}{lcl} \textbf{Case } 1 & \Leftrightarrow & TB(p-2)_M \leq k, \\ \textbf{Case } w & \Leftrightarrow & TB(p-w-1)_M \leq k < TB(p-w)_M \\ & & \quad \text{for } w = 2, 3, \cdots, p-1, \\ \textbf{Case } p & \Leftrightarrow & k < TB(0)_M. \end{array} \qquad (5.4)$$

The initial value of the cost of $\vec{a}^{(N)}$ denoted as $AC(N) = [A(h,i)_N]$, where $N = p^n$ and

$$A(h,i)_N = \begin{cases} 0 & if \quad h = 0, \\ 1 & if \quad h \neq 0 \end{cases} \qquad (5.5)$$

is obvious from the definition. I can compute $BC(M)$ and $AC(M)$ from $AC(pM)$ in the following way for $M = p^{n-1}, p^{n-2}, \cdots, p^0$ in the order. The cost of $\vec{b}(u)^{(M)}$ denoted as $BC(M) = [B(u,i)_M]$ can be computed from the previous cost $AC(pM) = [A(h,i)_{pM}]$ by

$$B(u,i)_M = \min\left\{ \sum_{j=0}^{p-1} A(e_j, jM+i)_{pM} \,\middle|\, \vec{e} \in D(u,i)_M \right\}, \tag{5.6}$$

where $\vec{e} = (e_0, e_1, \cdots, e_{p-1}) \in [GF(q)]^p$ and

$$D(u,i)_M = \{\vec{e} | F_s(e_0, e_1, \cdots, e_{p-1}) + b(s)_i^{(M)} = 0, \quad s = 0, 1, \cdots, u\} \tag{5.7}$$

is the set of all the $\vec{e}$'s which can make

$$b(0)_i^{(M)} = b(1)_i^{(M)} = \cdots = b(u)_i^{(M)} = 0. \tag{5.8}$$

The computation of $AC(M)$ from $AC(pM)$ depends on the case chosen at step (ii) of the generalized Games-Chan algorithm. Here I will introduce the "shift" and the "offset" in computing $AC(M)$. The shift is used to make

$$A(0,i)_M \le A(h,i)_M \quad \text{for } h = 1, 2, \cdots, q-1; i = 0, 1, \cdots, M-1. \tag{5.9}$$

The offset defined as

$$A(h,i)_M \leftarrow A(h,i)_M - A(0,i)_M \tag{5.10}$$

for $h = 0, 1, \cdots, q-1$; $i = 0, 1, \cdots, M-1$ is used to make $A(0,i)_M = 0$ for $i = 0, 1, \cdots, M-1$ after the shift is executed.

If I can force **Case** w to happen then I record the change vector of $\vec{a}^{(M)}$ as pM-tuple

$$\vec{c}_0(M) = (C(0,0)_M, C(0,1)_M, \cdots, C(0,pM-1)_M), \tag{5.11}$$

such that

$$\sum_{j=0}^{p-1} A(C(0,jM+i)_M, jM+i)_{pM} = B(p-w-1,i)_M \tag{5.12}$$

if $w = 1, 2, \cdots, p-1$ and

$$\begin{aligned} &\sum_{j=0}^{p-1} A(C(0,jM+i)_M, jM+i)_{pM} \\ &\quad = \min\left\{ \sum_{j=0}^{p-1} A(e_j, jM+i)_{pM} \,\middle|\, \vec{e} \in [GF(q)]^p \right\} \end{aligned} \tag{5.13}$$

if $w = p$, i.e. $C(0, jM+i)_M$ gives a change value of $a^{(pM)}_{jM+i}$ with the minimum number of changes in the initial N-tuple $\vec{a}^{(N)}$ under the condition that forcing **Case** w to happen is not altered at step (ii) of the generalized Games-Chan algorithm. I also compute the change matrix $C(M) = [C(h,i)_M]$ such that

$$A(h,i)_M = \sum_{j=0}^{p-1} A(C(h, jM+i)_M, jM+i)_{pM}. \tag{5.14}$$

The shift is performed by changing the possible region $D(h,i)^w_M$ of $\vec{e}$ used in computing

$$A(h,i)_M = \min\left\{ \sum_{j=0}^{p-1} A(e_j, jM+i)_{pM} \,\middle|\, \vec{e} \in \hat{D}(h,i)^w_M \right\} \tag{5.15}$$

by changing as

$$\hat{D}(h,i)^w_M = \left\{ \vec{e} \,\middle|\, \begin{array}{l} F_s(e_0,\cdots,e_{p-1}) + b(s)^{(M)}_i = 0 \\ \quad s = 0,1,\cdots,p-w-1 \\ F_{p-w}(e_0,\cdots,e_{p-1}) \\ \quad = F_{p-w}(C(0,i)_M,\cdots,C(0,(p-1)M+i)_M) + \alpha_h \end{array} \right\} \tag{5.16}$$

if $w = 1,2,\cdots,p-1$ and

$$\hat{D}(h,i)^p_M = \left\{ \vec{e} \,\middle|\, \begin{array}{l} F_0(e_0,\cdots,e_{p-1}) \\ \quad = F_0(C(0,i)_M,\cdots,C(0,(p-1)M+i)_M) + \alpha_h \end{array} \right\} \tag{5.17}$$

if $w = p$ and

$$\vec{a}^{(M)} \leftarrow F_{p-w}(\vec{a}(0)^{(pM)} + \vec{c}(0)^{(M)},\cdots,\vec{a}(p-1)^{(pM)} + \vec{c}(p-1)^{(M)}), \tag{5.18}$$

where $\vec{c}(j)^{(M)} = (C(0,jM)_M, C(0,jM+1)_M,\cdots,C(0,(j+1)M-1)_M)$ for $j = 0,1,\cdots,p-1$. Note that in the last relation of the definition of $\hat{D}(h,i)^w_M$, $\alpha_h + F_{p-w}(\vec{c}_0(M))$ is used instead of α_h. Since I have $\vec{c}_0(M) \in \hat{D}(0,i)^w_M$ and $\vec{c}_0(M) \notin \hat{D}(h,i)^w_M$ for $h \neq 0$, I have (5.9) from (5.12),(5.13) and (5.6),(5.7).

In the case of $w \neq p$ I have shown $B(p-w-1,i)_M = A(0,i)_M$ from (5.6) and (5.16). Therefore

$$\sum_{i=0}^{M-1} A(0,i)_M = TB(p-w-1)_M \leq k \qquad (5.19)$$

from (5.3) and (5.4). Therefore in order to keep the condition until the step M, I need at least (5.19) changes in the initial N-tuple $\vec{a}^{(N)}$. After the offset of $AC(M)$, I replace k by $k - TB(p-w-1)_M$.

Finally in order to determine an error vector I compute the error matrix at $\vec{a}^{(M)}$ as

$$E(M) = [E(h,i)_M], 0 \leq h \leq q-1; 0 \leq i \leq N-1 = p^n - 1. \quad (5.20)$$

The definition of $E(h,i)_M$ is as follows. In order to change $a_i^{(M)}$ $(0 \leq i \leq M-1)$ by α_h under the condition that the happening of **Case** w at step M is not altered, I change N/M elements $a_{jM+i}^{(N)}$ $(0 \leq j \leq \frac{N}{M} - 1)$ by $E(h, jM+i)_M$ in the original sequence $\vec{a}^{(N)}$ of length $N = p^n$.

Since such a change of $a_i^{(M)}$ by α_h can be obtained by changing $a_{jM+i}^{(pM)}$ $(0 \leq j \leq p-1)$ by $C(h, jM+i)_M$ from (5.14), I have

$$E(h, p\ell M + jM + i)_M = E(C(h, jM+i)_M, p\ell M + jM + i)_{pM} \quad (5.21)$$

for $h = 0, 1, \cdots, q-1$; $j = 0, 1, \cdots, p-1$; $\ell = 0, 1, \cdots, \frac{N}{pM} - 1$; $i = 0, 1, \cdots, M-1$. The initial value of the error matrix $E(N) = [E(h,i)_N]$, where $E(h,i)_N = \alpha_h$ for $i = 0, 1, \cdots, N-1$ is obvious from the definition.

5.3 The proposed Algorithm for Computing the k-LC

After above preparation, I propose the generalized Stamp-Martin algorithm with computing an error vector.

[**Generalized Stamp-Martin Algorithm**]

(i) Initial values: $N = pM = p^n$, $k\text{-}LC = 0$,
$\vec{a}^{(N)} = (a_0^{(N)}, a_1^{(N)}, \cdots, a_{N-1}^{(N)})$,

$$AC(N) = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}, \quad E(N) = \begin{bmatrix} \alpha_0 & \alpha_0 & \cdots & \alpha_0 \\ \alpha_1 & \alpha_1 & \cdots & \alpha_1 \\ \vdots & \vdots & & \vdots \\ \alpha_{q-1} & \alpha_{q-1} & \cdots & \alpha_{q-1} \end{bmatrix}.$$

(ii) Repeat the following 0)$\sim$2) until $M = 1$.

0). Compute $BC(M)$ and $TB(0)_M, \cdots, TB(p-2)_M$ by (4.6), (4.7), (5.2), (5.3) and (5.6).

1). Choose one of the following p cases.
Case 1: if $TB(p-2)_M \leq k$,
Case w: if $TB(p-w-1)_M \leq k < TB(p-w)_M$ for $w = 2, 3, \cdots, p-1$,
Case p: if $k < TB(0)_M$, respectively.

2). If **Case** w is chosen, then compute $\vec{a}^{(M)}$ by (5.18) and $k\text{-}LC \leftarrow k\text{-}LC + (w-1)M$.
If $w \neq p$ then $k \leftarrow k - TB(p-w-1)_M$.
First consider the shift operation by computing $AC(M)$ and $C(M)$ from (5.15)-(5.17) and (5.14).
Next consider the off-set operation (5.10) by computing $AC(M) = [A'(h,i)_M]$, which is the final $AC(M)$ at the step M.
Compute $E(M)$ by (5.21) from $E(pM)$ and $C(M)$.
If $M \neq 1$ then $M \leftarrow M/p$ and go to 0).

(iii). Let $\vec{a}^{(1)} = (a_0^{(1)}) = (\alpha_s)$, $\alpha_t = -\alpha_s$.
If $A(t,0)_1 > k$ then $k\text{-}LC \leftarrow k\text{-}LC{+}1$ and $\vec{e} = (E(0,0)^{(1)}, E(0,1)^{(1)}, \cdots, E(0, N-1)^{(1)})$,
else $\vec{e} = (E(t,0)^{(1)}, E(t,1)^{(1)}, \cdots, E(t, N-1)^{(1)})$.

The final $k\text{-}LC$ is equal to k-LC($\{a_i\}$) and $\vec{e}$ is an error vector.□

In the proposed algorithm with the shift and offset of the cost, I do not need to record the matrices $AC(M)$, $BC(M)$, $C(M)$ and $E(M)$ after the next matrices at the step M/p is calculated by their matrices at the step M since the vector $\vec{a}^{(M)}$ is changed into the vector satisfied the minimum number of changes and the condition until the step M at every steps. Moreover the first row of $AC(M)$ can be forgot, i.e. all of these is zero at any time. Therefore the proposed algorithm needs only one step for the k-LC and an error vector, and less memory than the algorithm in [KUI99].

5.4 A Numerical Example

In this section I show a small example for 3-LC, i.e. $k = 3$, of a sequence over $GF(3^2)$ with period 3^3. Let $\{a_i\}$ be a sequence over

$$GF(3^2) = \{\alpha_0 = \alpha^* = 0, \alpha_1 = \alpha^0 = 1, \alpha_2 = \alpha^1, \cdots, \alpha_8 = \alpha^7\} \tag{5.22}$$

with period $3^3 = 27$, where α is a primitive of $GF(9)$ such that $\alpha^2 = 2\alpha + 1$. I will denote α^*, α^i as $*$, i, respectively for simplicity. Let one period of $\{a_i\}$ be

$$\vec{a}^{(27)} = (012\ 345\ 670\ 012\ 345\ 671\ 012\ 345\ 672). \tag{5.23}$$

I will compute 3-LC ($k = 3$) and an error vector as follows.

[Example 5.1]

(initial values)

$\vec{a}^{(27)} = (012\ 345\ 670\ 012\ 345\ 671\ 012\ 345\ 672)$

$$AC(27) = \begin{bmatrix} 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 \\ 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 \\ 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 \\ 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 \\ 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 \\ 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 \\ 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 \\ 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 \\ 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 \end{bmatrix}$$

$$E(27)=\begin{bmatrix} *** & *** & *** & *** & *** & *** & *** & *** & *** \\ 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 \\ 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 & 111 \\ 222 & 222 & 222 & 222 & 222 & 222 & 222 & 222 & 222 \\ 333 & 333 & 333 & 333 & 333 & 333 & 333 & 333 & 333 \\ 444 & 444 & 444 & 444 & 444 & 444 & 444 & 444 & 444 \\ 555 & 555 & 555 & 555 & 555 & 555 & 555 & 555 & 555 \\ 666 & 666 & 666 & 666 & 666 & 666 & 666 & 666 & 666 \\ 777 & 777 & 777 & 777 & 777 & 777 & 777 & 777 & 777 \end{bmatrix}$$

$(M=9)$

$\vec{b}(0)^{(9)}=(*** \ *** \ **4)$ $\quad \vec{b}(1)^{(9)}=(*** \ *** \ **6)$

$$BC(9)=\begin{bmatrix} 000 & 000 & 001 & TB(0)_9=1 \\ 000 & 000 & 002 & TB(1)_9=2 \end{bmatrix}$$

I choose **Case 1**, since $TB(1)_9=2\leq k=3$.

$\vec{a}^{(9)}=(012\ 345\ 670)$

$k\text{-}LC\leftarrow 0 \quad k\leftarrow 3-2=1$

$$AC(9)=\begin{bmatrix} 000 & 000 & 000 \\ 333 & 333 & 331 \\ 333 & 333 & 331 \\ 333 & 333 & 331 \\ 333 & 333 & 331 \\ 333 & 333 & 331 \\ 333 & 333 & 330 \\ 333 & 333 & 330 \\ 333 & 333 & 331 \end{bmatrix}$$

$$E(9)=\begin{bmatrix} *** & *** & *** & *** & *** & **2 & *** & *** & **1 \\ 000 & 000 & 000 & 000 & 000 & 003 & 000 & 000 & 007 \\ 111 & 111 & 111 & 111 & 111 & 110 & 111 & 111 & 115 \\ 222 & 222 & 222 & 222 & 222 & 226 & 222 & 222 & 220 \\ 333 & 333 & 333 & 333 & 333 & 331 & 333 & 333 & 334 \\ 444 & 444 & 444 & 444 & 444 & 445 & 444 & 444 & 446 \\ 555 & 555 & 555 & 555 & 555 & 557 & 555 & 555 & 55* \\ 666 & 666 & 666 & 666 & 666 & 66* & 666 & 666 & 663 \\ 777 & 777 & 777 & 777 & 777 & 774 & 777 & 777 & 772 \end{bmatrix}$$

$(M=3)$

$\vec{b}(0)^{(3)}=(4\ 5\ 6)$ $\quad \vec{b}(1)^{(3)}=(2\ 3\ 4)$

$$BC(3) = \begin{bmatrix} 3 & 3 & 1 & TB(0)_3 = 7 \\ 6 & 6 & 3 & TB(1)_3 = 15 \end{bmatrix}$$

I choose **Case 3**, since $k = 1 < TB(0)_3 = 7$.

$\vec{a}^{(3)} = (4\ 5\ 6)$

$k\text{-}LC \leftarrow 0 + 2 \times 3 = 6 \quad k \leftarrow 1$

$$AC(3) = \begin{bmatrix} 0 & 0 & 0 \\ 3 & 3 & 1 \\ 3 & 3 & 1 \\ 3 & 3 & 1 \\ 3 & 3 & 1 \\ 3 & 3 & 1 \\ 3 & 3 & 0 \\ 3 & 3 & 0 \\ 3 & 3 & 1 \end{bmatrix}$$

$$E(3) = \begin{bmatrix} *** & *** & *** & *** & *** & **2 & *** & *** & **1 \\ *** & *** & 000 & *** & *** & 003 & *** & *** & 007 \\ *** & *** & 111 & *** & *** & 110 & *** & *** & 115 \\ *** & *** & 222 & *** & *** & 226 & *** & *** & 220 \\ *** & *** & 333 & *** & *** & 331 & *** & *** & 334 \\ *** & *** & 444 & *** & *** & 445 & *** & *** & 446 \\ *** & *** & 555 & *** & *** & 557 & *** & *** & 55* \\ *** & *** & 666 & *** & *** & 66* & *** & *** & 663 \\ *** & *** & 777 & *** & *** & 774 & *** & *** & 772 \end{bmatrix}$$

$(M = 1)$

$\vec{b}(0)^{(1)} = (0) \quad \vec{b}(1)^{(1)} = (2)$

$$BC(1) = \begin{bmatrix} 1 & TB(0)_1 = 1 \\ 3 & TB(1)_1 = 3 \end{bmatrix}$$

I choose **Case 2**, since $TB(0)_1 = 1 \leq k = 1 < TB(1)_1 = 3$.

$\vec{a}^{(1)} = (2) \quad k\text{-}LC \leftarrow 6 + 1 = 7 \quad k \leftarrow 1 - 1 = 0$

$$AC(1) = \begin{bmatrix} 0 \\ 2 \\ 2 \\ 2 \\ 3 \\ 2 \\ 2 \\ 2 \\ 3 \end{bmatrix}$$

$$E(1) = \begin{bmatrix} *** & *** & **4 & *** & *** & **5 & *** & *** & **6 \\ *** & *** & 4** & *** & *** & 4*2 & *** & *** & 4*1 \\ *** & *** & 5*6 & *** & *** & 5** & *** & *** & 5*3 \\ *** & *** & 6*5 & *** & *** & 6*7 & *** & *** & 6** \\ *** & *** & *31 & *** & *** & *30 & *** & *** & *35 \\ *** & *** & *4* & *** & *** & *42 & *** & *** & *41 \\ *** & *** & *56 & *** & *** & *5* & *** & *** & *53 \\ *** & *** & *65 & *** & *** & *67 & *** & *** & *6* \\ *** & *** & *72 & *** & *** & *76 & *** & *** & *70 \end{bmatrix}$$

Since $\vec{a}^{(1)} = (\alpha^2)$, $-\alpha^2 = \alpha^7$ and $A(8,0)_1 = 3 > k = 0$, then k-$LC \leftarrow 7 + 1 = 8$ and an error vector $\vec{e}$ is the first row of $E(1)$. Hence k-$LC(\{a_i\}) = 8$ and

$$\vec{e} = (*** \ *** \ **4 \ *** \ *** \ **5 \ *** \ *** \ **6). \quad (5.24)$$

5.5 Conclusion

The new algorithm using the shift and offset of the cost matrix $AC(M)$ in this paper is a generalization of the Stamp-Martin algorithm[SM93], since in case of binary sequences I can show that the part of computing the k-LC in the proposed algorithm is the same as the Stamp-Martin algorithm.

I am interested in more extension about the period of sequences, i.e., considering sequences over $GF(p^m)$ with period $N = N_0 p^n$, where $\gcd(N_0, p) = 1$. Blackburn[Bla94] gave an algorithm for the LC of general periodic sequences using the generalized Games-Chan algorithm and the discreet Fourier transform.

Chapter 6

Complexity Analysis of the Algorithms

6.1 Introduction

Unfortunately the complexity analysis of the algorithms has not studied yet[KUI98-3]. In this chapter I try to evaluate the time-complexity and the space-complexity in order to compare with the generalized Stamp-Martin algorithm in Chapter 5 and the generalized k-LC algorithm in Chapter 4.

One of the algorithms is the generalized Stamp-Martin algorithm with the shift and offset of the cost, shown in chapter 5. The generalized Stamp-Martin algorithm has only one step to decide the k-LC and an error vector. Therefore I can calculate the k-LC and the error vector at the same time.

Another algorithm is the generalized k-LC algorithm without the shift and offset of the cost, shown chapter 4. The generalized k-LC algorithm consists two steps in order to decide the k-LC and the error vector. Therefore I know the error vector after the calculating the k-LC.

In Section 6.2 I rewrite the generalized k-LC algorithm in order to evaluate the time-complexity and the space-complexity easily. Next the different points of the generalized Stamp-Martin algorithm from the generalized k-LC algorithm in this section are shown.

In section 6.3 I discuss the time-complexity of two algorithms for

k-LC and an error vector of sequences $GF(p^m)$ with period p^n using the algorithms in Section 6.2. However the time-complexity in this section evaluate only the number of the operations about addition and subtraction.

In section 6.4 I discuss the space-complexity of two algorithms for k-LC and an error vector of sequences $GF(p^m)$ with period p^n using the algorithms in Section 6.2.

6.2 The algorithms for Computing the k-LC

In order to compare two algorithms for k-LC I write these again.

[generalized k-LC algorithm]

(i) Initial Values: $N = pM = p^n, \quad k\text{-LC}= 0,$

$\vec{a}^{(N)} = (a_0^{(N)}, a_1^{(N)}, \cdots, a_{N-1}^{(N)}),$

$$AC(N) = [A(h,i)_N] = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

(ii) Repeat (0)–(6) until $M = 1$.

(0) Compute $BC(M) = [B(u,i)_M]$ for $(0 \le u \le p-2,\ 0 \le i \le M-1)$ by

$$B(u,i)_M = \min\left\{ \sum_{j=0}^{p-1} A(e_j, i+jM)_{pM} \,\middle|\, \vec{e} \in D(u,i)_M \right\}, \tag{6.1}$$

where $\vec{e} = (e_0, e_1, \cdots, e_{p-1}) \in [GF(q)]^p$,

$$D(u,i)_M = \left\{ \vec{e} \,\middle|\, \begin{array}{l} F_j(e_0, e_1, \cdots, e_{p-1}) + b_{j,i}^{(M)} = 0 \\ (0 \le j \le u) \end{array} \right\}, \tag{6.2}$$

$$\begin{aligned}\vec{b}(u)^{(M)} &= (b_{u,0}\,(M), b_{u,1}\,(M), \cdots, b_{u,M-1}\,(M)) \\ &= F_u(\vec{a}(0)^{(pM)}, \vec{a}(1)^{(pM)}, \cdots, \vec{a}(p-1)^{(pM)}) \\ &= \sum_{j=0}^{p-u-1} c_{u,j}\vec{a}(j)^{(pM)}\end{aligned} \tag{6.3}$$

and

$$c_{u,j} = \begin{pmatrix} p-j-1 \\ u \end{pmatrix}. \tag{6.4}$$

(1) Compute $TB(0)_M, TB(1)_M, \cdots, TB(p-2)_M$ from

$$TB(u)_M = \sum_{i=0}^{M-1} B(u,i)_M. \tag{6.5}$$

(2) Choose only one Case w in p cases as Case 1 $\sim$ Case p from

$$\begin{cases} \text{Case } 1 & \Leftarrow \;\; TB(p-2)_M \le k \\ \text{Case } w & \Leftarrow \;\; TB(p-w-1)_M \le k < TB(p-w)_M \\ \text{Case } p & \Leftarrow \;\; k < TB(0)_M \end{cases} \tag{6.6}$$

(a) (3) If Case w is chosen, then compute $\vec{a}^M$ and k-LC from

$$\vec{a}^{(M)} = F_{p-w}(\vec{a}(0)^{(pM)}, \vec{a}(1)^{(pM)}, \cdots, \vec{a}(p-1)^{(pM)}) \tag{6.7}$$

and

$$k\text{-}LC \leftarrow k\text{-}LC + (w-1)M. \tag{6.8}$$

(4) compute $AC(M) = [AC(h,i)_M]$ for $0 \le h \le q-1,\ 0 \le i \le M-1$ from

$$A(h,i)_M = \min\left\{ \sum_{j=0}^{p-1} A(e_j, i+jM)_{pM} \,\middle|\, \vec{e} \in \hat{D}(h,i)^w_M \right\}, \tag{6.9}$$

where

$$\hat{D}^w_M = \left\{ \vec{e} \,\middle|\, \begin{array}{l} F_j(e_0, e_1, \cdots, e_{p-1}) + b^{(M)}_{j,i} = 0 \\ (0 \le j \le p-2) \\ F_{p-w}(e_0, e_1, \cdots, e_{p-1}) - \alpha_h = 0 \end{array} \right\} \tag{6.10}$$

if $1 \leq w \leq p-1$ or

$$\hat{D}_M^p = \{\vec{e} \mid F_0(e_0, e_1, \cdots, e_{p-1}) - \alpha_h = 0\} \tag{6.11}$$

if $w = p$.

(5) Let $\vec{e}(h,i)_M$ in $E(M) = [\vec{e}(h,i)_M]$ for $0 \leq h \leq q-1$, $0 \leq i \leq M-1$ be $\vec{e}$ in right hand side of (6.9) when $A(h,i)_M$ is calculated at the step (4).

(6) If $M \neq 1$, then let $M \leftarrow M/p$ and go to (0).

(iii) If $A(-a_0^{(1)}, 0)_1 > k$, then $k\text{-}LC \leftarrow k\text{-}LC + 1$

(iv) Let

$$\begin{cases} A(-a_0^{(1)}, 0)_1 \leq k & \Rightarrow \quad \vec{e}(1) = \vec{e}(-a_0^{(1)}, 0)_1, \\ A(-a_0^{(1)}, 0)_1 > k & \Rightarrow \quad \vec{e}(1) = \vec{e}(s, 0)_1, \end{cases}$$

where s is such that $A(s,0)_1 = \min\{A(h,0)_1 \mid 0 \leq h \leq q-1\}$.

(v) Compute $\vec{e}(pM) = (\vec{e}_0^{(pM)}, \vec{e}_1^{(pM)}, \cdots, \vec{e}_{pM-1}^{(pM)})$ from $\vec{e}(M) = (\vec{e}_0^{(M)}, \vec{e}_1^{(M)}, \cdots, \vec{e}_{M-1}^{(M)})$ and $E(pM)$ by

$$\vec{e}_{i+jM}^{(pM)} = \vec{e}(e_{i,j}, i+jM)_{pM} \quad (0 \leq j \leq p-1,\ 0 \leq i \leq M-1), \tag{6.12}$$

where $\vec{e}_i^{(M)} = (e_{i,0}, e_{i,1}, \cdots, e_{i,p-1})$.

(vi) Let $M \leftarrow pM$.
If $M \neq p^{n-1}$, then go to (v).

(vii) Compute $\vec{e} = (e_0, e_1, \cdots, e_{N-1})$ by

$$e_{i+jM} = e_{i,j} \quad (0 \leq i \leq M-1,\ 0 \leq j \leq p-1). \tag{6.13}$$

The steps from (i) to (iii) give k-LC and the steps form (iv) to (vii) give an error vector at above algorithm.

Next I show the generalized Stamp-Martin algorithm which gives the k-LC and the error vector at the same time. The differences from the generalized k-LC algorithm are the size of $q \times N$ and the element over $GF(q)$ with the respect to the error matrix $E(M)$. At the generalized k-LC algorithm the size of $E(M)$ is $q \times M$ and the element is over $[GF(q)]^p$.

Moreover $q \times M$ the change matrix $C(M) = [C(h,i)_M]$ is defined as $C(h,i)_M$ such that

$$A(h,i)_M = \sum_{j=0}^{p-1} A(C(h,i+jM)_M, i+jM)_{pM}. \tag{6.14}$$

In (6.9) I define $\hat{D}(h,i)_M^w$ also by

$$\hat{D}_M^w = \left\{ \vec{e} \left| \begin{array}{l} F_j(e_0,e_1,\cdots,e_{p-1}) + b_{j,i}^{(M)} = 0 \quad (0 \leq j \leq p-w-1) \\ F_{p-w}(e_0,e_1,\cdots,e_{p-1}) \\ \quad = F_{p-w}(C(0,i)_M,\cdots,C(0,i+(p-1)M)) + \alpha_h \end{array} \right. \right\} \tag{6.15}$$

if $1 \leq w \leq p-1$ or

$$\hat{D}_M^p = \left\{ \vec{e} \left| \begin{array}{l} F_0(e_0,e_1,\cdots,e_{p-1}) \\ \quad = F_0(C(0,i)_M,\cdots,C(0,i+(p-1)M)) + \alpha_h \end{array} \right. \right\}, \tag{6.16}$$

if $w = p$. And computing $\vec{a}^{(M)}$ defined by

$$\vec{a}^{(M)} = F_{p-w}(\vec{a}(0)^{(pM)} + \vec{c}(0)^{(M)}, \cdots, \vec{a}(p-1)^{(pM)} + \vec{c}(p-1)^{(M)}), \tag{6.17}$$

where

$$\vec{c}(j)^{(M)} = (C(0,jM)_M, \cdots, C(0,(j+1)M-1)_M). \tag{6.18}$$

The shift of $AC(M)$ is executed after computing $AC(M)$ as follows. After

$$\begin{array}{l} A(h,i)'_M = A(h,i)_M - A(0,i)_M \\ (0 \leq h \leq q-1,\ 0 \leq i \leq M-1) \end{array} \tag{6.19}$$

is computed, let $AC(M)$ into $AC(M) = [A(h,i)'_M]$. Finally k is changed to $k \leftarrow TB(p-w-1)_M$.

I explain the method for the error vector. I set initial value of $E(N)$ by

$$E(N) = \begin{bmatrix} \alpha_0 & \alpha_0 & \cdots & \alpha_0 \\ \alpha_1 & \alpha_1 & \cdots & \alpha_1 \\ \vdots & \vdots & & \vdots \\ \alpha_{q-1} & \alpha_{q-1} & \cdots & \alpha_{q-1} \end{bmatrix} \tag{6.20}$$

At each step I compute $q \times N$ matrix $E(M) = [E(h,i)_M]$ from $E(pM)$ by

$$E(h, p\ell M+i+jM)_M = E(C(h, i+jM)_M, p\ell M+i+jM)_{pM} \quad (6.21)$$

from $(0 \leq h \leq q-1,\ 0 \leq j \leq p-1,\ 0 \leq \ell \leq \frac{N}{pM}-1,\ 0 \leq i \leq M-1)$. Finally an error vector $\vec{e}$ is decided by

$$\vec{e} = (E(0,0)_1, E(0,1)_1, \cdots, E(0, N-1)_1) \quad (6.22)$$

if $A(t,0)_1 > k$ or

$$\vec{e} = (E(t,0)_1, E(t,1)_1, \cdots, E(t, N-1)_1) \quad (6.23)$$

if $A(t,0)_1 \leq k$ with respect to $E(1)$ at the final step, where t is satisfying $\alpha_t = -a_0^{(1)}$.

6.3 Evaluation of the Time-complexity

In this section I discuss evaluation of the time-complexity for the two algorithms. I count the number of additions and subtractions the worst case with respect to the algorithms in order to evaluate the time-complexity.

Firstly I consider same number of operations between the generalized k-LC algorithm and the generalized Stamp-Martin algorithm. The number of computing $BC(M)$ at the step m is $p\dfrac{p^{m(p-1)} - p^m}{p^m - 1} \times M$, since the cardinality of $D(u,i)_M$ is $p^{m(p-u-1)}$ depending on u and the number par one error vector is $p \times M$, where p is the length of the error vector and M is the number of positions or the size of column with respect to $BC(M)$. Next the number of computing $TB(u)$ at the step M is $(p-1) \times M$ with respect to the both of two algorithms.

Secondly I consider the difference between two algorithms. The number of computing $AC(M)$ at the step M with respect to the generalized Stamp-Martin algorithm is $p^{(p-1)m+1} \times M$ and The number of computing $AC(M)$ at the step M with respect to the generalized k-LC algorithm is $p^{pm+1} \times M$ since the cardinality of $\hat{D}(u,i)_M$ at the generalize k-LC algorithm is p^m times of the cardinality of $\hat{D}(u,i)_M$ at the generalized Stamp-Martin algorithm.

Finally I need $p^n \times M$ times at the offset of the cost and n times at changing k only the generalized Stamp-Martin algorithm.

Here the values of M take $p^{n-1}, p^{n-2}, \cdots, p^0 = 1$ at the step M in the order. Therefore since the summation of M's is $\dfrac{p^n - 1}{p-1}$, I have the same order $O(p^{(p-1)m+n})$ about the time-complexity with respect to the both of two algorithms. However I know the generalized Stamp-Martin algorithm has nearly 1/2 times coefficient at $p^{(p-1)m+n}$ to the generalized k-LC algorithm.

6.4 Evaluation of the Space-complexity

In this section I discuss evaluation of the space-complexity for the two algorithms. I count the number of required memories with respect to the algorithms in order to evaluate the space-complexity.

Firstly I consider same number of memories between the generalized k-LC algorithm and the generalized Stamp-Martin algorithm. From variables M and k-LC I need 4 including temporary variables M' and k' at changing. The number of $\vec{a}$ is $(p+1)p^{n-1}$ and the number of $AC(M)$ is $p^m(p+1)p^{n-1}$ including temporary variables. Since variables $BC(M)$ and $TB(0), \cdots, TB(p-2)$ do not need temporary variable, the number of $BC(M)$ and $TB(0), \cdots, TB(p-2)$ are $(p-1)p^{n-1}$ and $p-1$, respectively.

Secondly I consider the difference between two algorithms. With respect to k the generalized k-LC algorithm does not need temporary variable but the generalized Stamp-Martin algorithm does need. Hence the generalized k-LC algorithm and the generalized Stamp-Martin algorithm need 1 and 2 about k, respectively. Next with respect to $E(M)$, the generalized k-LC algorithm needs to store all of $E(N)$, $E(N/p)$, $\cdots$, $E(1)$. On the other hand the generalized Stamp-Martin algorithm needs only one $E(N)$ at some step and temporary variable for changing. Therefore the numbers of $E(N)$, $E(N/p)$, $\cdots$, $E(1)$ are $p^{m+1}\sum_{i=0}^{n-1} p^i = p^{m+1}\dfrac{p^n-1}{p-1}$ and $2p^{m+n}$ with respect to the generalized k-LC algorithm and the generalized

Stamp-Martin algorithm, respectively. Finally the generalized k-LC algorithm needs $(p+1)p^{n-1}$ with respect to $\vec{e}$ including temporary variable and the generalized Stamp-Martin algorithm needs p^{m+n} with respect to $C(M)$.

Therefore I have also the same order $O(p^{m+n})$ about the space-complexity with respect to the both of two algorithms. However I know the generalized k-LC algorithm has nearly 2/3 times coefficient at p^{m+n} to the generalized Stamp-Martin algorithm.

6.5 conclusion

In this section I show the different points between the generalized Stamp-Martin algorithm and the generalized k-LC algorithm. In order to evaluate the time-complexity and the space-complexity, I count the number of operations as addition and subtraction, and required memories.

It become clear that the time and space-complexity for the generalized Stamp-Martin algorithm and the generalized k-LC algorithm have same order complexity, respectively. In view of the coefficient at the maximum order, the generalized Stamp-Martin algorithm is superior about the time-complexity and inferior about the space-complexity.

It remains that I consider the comparison and the setting variables when the time-complexity is evaluating. Many procedures at the part of computing the error vector are shared by the comparison and the setting variables.

Chapter 7

Conclusions and Future Works

7.1 Conclusions

The main result in this dissertation is the generalized algorithms for the k-LC and the error vector of sequences over $GF(p^m)$ with period p^n.

Firstly I consider sequences over $GF(2)$ with period 2^n. In this case the Stamp-Martin algorithm has been given by M. Stamp and C. F. Martin[SM93]. However another algorithm for the k-LC with modified cost matrix not but the cost vector in the Stamp-Martin algorithm. Moreover this algorithm decides not only the k-LC but also the error vector which gives the k-LC. I show that the proposed algorithm is equivalent to the Stamp-Martin algorithm by the Proposition 2.1. Two examples are given at the end of Chapter 2.

Secondly the algorithm for the k-LC of sequences over $GF(3)$ with period 3^n is given in Chapter 3. This generalization of the algorithm into sequences over $GF(3)$ with period 3^n is derived by the case of $p = 3$ and $m = 1$ in the generalized Games-Chan algorithm for the LC of sequences over $GF(p^m)$ with period p^n. Hence this result leads to more generalization into sequences over $GF(p^m)$ with period p^n as the first step. In the end of Chapter 3 the performance of the proposed algorithm and the profile of the k-LC about k is shown as Example 3.1. However the concepts of the shift and offset of the cost are not used in that chapter.

Thirdly the algorithm for the k-LC and the error vector of sequences over $GF(p^m)$ with period p^n is shown in Chapter 4. This algorithm is derived by the generalized Games-Chan algorithm similar to the case of $p = 3$ and $m = 1$ in Chapter 3. However the concepts of the shift and offset of the cost are not used also in that chapter. The algorithm using the concepts of the shift and offset of the cost is shown in the next chapter.

In the same as the case of binary sequences the concepts of the shift and offset of the cost can derive another algorithm for the k-LC and the error vector of sequences over $GF(p^m)$ with period p^m. In Chapter 5 another algorithm for the k-LC and the error vector of sequences over $GF(p^m)$ with period p^n is showed by the shift and offset of the cost. Because of the shift and offset the proposed algorithm needs only one step to computing the k-LC and the error vector, i.e. the algorithm in Chapter 4 needs two steps consisted the first step computing the k-LC and the second step computing the error vector but the proposed algorithm in that chapter is not need two steps. An example is given also at the end of Chapter 5.

In Chapter 6 I discuss the time-complexity and the space-complexity about the algorithms in Chapter 4 and Chapter 5. The time-complexity is the number of the addition and subtraction and the space-complexity is the number of required variable. The algorithm in Chapter 4 is called the generalized k-LC algorithm and the algorithm in Chapter 5 is called the generalized Stamp-Martin algorithm. Two algorithm have same order about the time-complexity and the space-complexity. However in view of the coefficient of the maximum order, the generalized Stamp-Martin algorithm is superior nearly 1/2 to the generalized k-LC algorithm with respect to the time-complexity. Next in view of the coefficient of the maximum order, the generalized k-LC algorithm is superior nearly 2/3 to the generalized Stamp-Martin algorithm with respect to the space-complexity. I need to note that I do not consider the comparison and the setting variable in the time-complexity and the subscript in the space-complexity.

7.2 Future Works

In this section I discuss remaining problems about the k-LC and the algorithm for the k-LC and the error vector.

Firstly I think main problem is more generalization about the period of sequences. I know the fast algorithms for the k-LC and the error vector only case of sequences over $GF(p^m)$ with period p^n, i.e. I do not know fast algorithms for the k-LC and the error vector in the case of sequences over $GF(p^m)$ with period N_0p^n, where p is a prime and N_0 is coprime against a prime p. If an algorithm is given in this case, I know the algorithm for the k-LC and the error vector in any case. In 1994 S. R. Blackburn[Bla94] gave the algorithm for the LC in the case of sequences over $GF(p^m)$ with period N_0p^n. It seems that this result is useful to solve the above problem.

The algorithm given in this dissertation executes over a finite field. Generalization of ground set into finite rings[MUKI96] from finite fields is second problem remaining. The proposed algorithms need only the addition and the subtraction. Hence I do not need to care about the inverse element with respect to multiplication.

Next I would like to evaluate the time-complexity and the space-complexity of the proposed algorithms more strictly. In Chapter 6 I do not consider the operations of the comparison and the setting variable in the time-complexity and the subscript in the space-complexity. Hence the detail analysis about above problems may lead to make some difference between the two algorithms. In applications these evaluations are very important.

Thirdly modifications of the proposed algorithms is remained, for instance, if the profile of the k-LC is needed, you must apply one of the proposed algorithms for $k = 0$, $k = 1$, $\cdots$, $k = N$, where N is the period of given sequence. I think some reduction can be derived.

The proposed algorithms give only one error vector, but the error vector is not unique in general. If I do not need to consider about computational complexity, the algorithm for all error vectors can be derived by the stored data of the change value in all cases at all steps.

Finally, I do not discuss about the property of k-LC over any sequences. In future this problem is solved about the typical sequences. I think these results is useful for many fields such as random numbers, cryptography and communication systems. For instance, the profile of the k-LC about k may be very important in the theory or the applications.

Bibliography

[Berl68] E. R. Berlekamp, *Algebraic Coding Theory*, New York: McGraw-Hill, 1968.

[Bla94] S. R. Blackburn, "A generalisation of the discrete Fourier transform: determining the minimal polynomial of a periodic sequence", *IEEE Trans. Inform. Theory*, Vol. 40, pp.1702-1704, Sep. 1994.

[DI98] Z. Dai and K. Imamura, "Linear complexity for one-symbol substitution of a periodic sequence over $GF(q)$", *IEEE Trans. Inform. Theory*, Vol. 44, pp. 1328-1331, May 1998.

[DXS91] C. Ding, G. Xiao and W. Shan, W, *The Stability Theory of Stream Ciphers*, Lecture Notes in Computer Science, Vol. 561, Springer-Verlag 1991.

[GC83] R. A. Games and A. H. Chan, "A fast algorithm for determining the complexity of a binary sequence with period 2^n', *IEEE Trans. Inform. Theory*, Vol. IT-29, pp.144-146, Jan. 1983.

[IMU91] K. Imamura, T. Moriuchi and S. Uehara, "Periodic sequences of the maximum linear complexity simply obtained from an m-sequence", *Proc. IEEE 1991 Intern'l Symp. on Inform. Theory*, p.175, June 1991.

[IM93] K. Imamura and T. Moriuchi, "A fast algorithm for determining the linear complexity of p-ary sequence with period p^n, p a prime", *IEICE Tech. Rept.*, Vol. IT93-75, pp. 73-78, Dec. 1993 (in Japanese).

[IUM94] K. Imamura, S. Uehara and T. Moriuchi, "$GF(q)$ sequences of period $q^n - 1$ with maximum linear complexity and maximum order complexities for minimum changes of m-sequences", *Proc. IEEE 1994 Intern'l Symp. on Inform. Theory*, p.366, June 1994.

[KUI96] T. Kaida, S. Uehara and K. Imamura, "An algorithm for the k-error linear complexity of sequences over $GF(3)$ with period 3^n", *Proc. 1996 Intern'l Symp. Inform. Theory and Its Applications*, pp.155-158, Sep. 1996.

[KUI96-2] T. Kaida, S. Uehara and K. Imamura, "Computation of the k-error linear complexity of binary sequences with period 2^n", J. Jaffar and R.H.C. Yap (eds.), *Concurrency and Parallelism, Programming, Networking, and Security*, Lecture Notes in Computer Science, Vol. 1179, pp.182-191, Springer-Verlag, 1996.

[KUI98] T. Kaida, S. Uehara and K. Imamura, "An algorithm for the k-error linear complexity of sequences over $GF(p^m)$ with period p^n, p a prime", *Proc. 1998 Intern'l Symp. on Inform. Theory and Its Applications*, pp.686-689, Oct. 1998.

[KUI98-2] T. Kaida, S. Uehara and K. Imamura, "An new algorithm for the k-error linear complexity of sequences over $GF(p^m)$ with period p^n", accepted to the Intern'l Conference on Sequences and Their Applications (SETA'98), Dec. 1998.

[KUI98-3] T. Kaida, S. Uehara and K. Imamura, "Complexity analysis of algorithms for the k-error linear complexity", *Proc. the 21st Symp. Inform. Theory and Its Applications*, pp.443-446, Dec. 1998 (in Japanese).

[KUI99] T. Kaida, S. Ueharaand K. Imamura, "An algorithm for the k-error linear complexity of sequences over $GF(p^m)$ with period p^n, p a prime", to appear *Information and Computation*, Academic Press, May 1999.

[Mas69] J. L. Massey, "Shift register synthesis and BCH decoding", *IEEE Trans. Inform. Theory*, Vol. IT-15, pp.122-127, Jan. 1969.

[MS86] J. L. Massey and T. Schaub, "Linear complexity in coding theory", Lecture Notes in Computer Science, Vol. 331, pp.19-32, Springer-Verlag, 1986.

[Mil75] W. H. Mills, "Continued fractions and linear recurrences", *Math. of Computation*, Vol. 29, pp.173-180, 1975.

[MUKI96] T. Moriuchi, S. Uehara, T. Kaida and K. Imamura, "Some families of sequences over Z_4 and Z_8, and their characteristic polynomials", *Proc. 1996 Intern'l. Symp. on Inform. Theory and Its Applications*, pp.78-81, Sep. 1996.

[Rup86] R. A. Ruppel, "Linear complexity and random sequences", Lecture Notes in Computer Science, Vol. 219, Springer-Verlag, 1986.

[SM93] M. Stamp and C. F. Martin, "An algorithm for the k-error linear complexity of binary sequences with period 2^n", *IEEE Trans. Inform. Theory*, Vol. 39, pp.1398-1401, July 1993.

[UI96] S. Uehara and K. Imamura, "Linear complexity of periodic sequences obtained from $GF(q)$ sequences with period $q^n - 1$ by one-symbol insertion", *IEICE Trans.*, Vol. E79-A, pp.1739-1740, Oct. 1996.

[UI97] S. Uehara and K. Imamura, "Linear complexity of periodic sequences obtained from a sequence over $GF(p)$ with period $p^n - 1$ by one-symbol deletion", *IEICE Trans.*, Vol. E80-A, pp.1164-1166, June 1997.

[UIK97] S. Uehara, K. Imamura and T. Kaida, T, "Value distribution of linear complexity for p-ary periodic sequences with period p^n, p a prime", *IEICE Trans.*, Vol. E80-A, pp.920-921, May 1997.

List of Publications by the Author

Journal Papers

(1) T. Kaida, S. Uehara and K. Imamura, "Computation of the k-error linear complexity of binary sequences with period 2^n", J. Jaffar and R.H.C. Yap (eds.), *Concurrency and Parallelism, Programming, Networking, and Security*, Lecture Notes in Computer Science, Vol. 1179, pp.182-191, Dec. 1996.

(2) S. Uehara, K. Imamura and T. Kaida, "Value distribution of linear complexity for p-ary periodic sequences with period p^n, p a prime", *IEICE Trans. Fundamentals*, Vol. E80-A, pp.920-921, May 1997.

(3) T. Kaida, S. Uehara and K. Imamura, "An algorithm for the k-error linear complexity of sequences over $GF(p^m)$ with period p^n, p a prime", to appear *Information and Computation*, May 1999.

International Conference Papers

(1) T. Kaida, S. Uehara and K. Imamura, "An algorithm for the k-error linear complexity of sequences over $GF(3)$ with period 3^n", *Proc. 1996 Intern'l. Symp. on Information Theory and Its Applications*, pp.155-158, Victoria Canada, Sep. 1996.

(2) T. Moriuchi, S. Uehara, T. Kaida and K. Imamura, "Some families of sequences over Z_4 and Z_8, and their characteris-

tic polynomials", *Proc. 1996 Intern'l. Symp. on Information Theory and Its Applications*, pp.78-81, Victoria Canada, Sep. 1996.

(3) T. Kaida, S. Uehara and K. Imamura, "An algorithm for the k-error linear complexity of sequences over $GF(p^m)$ with period p^n, p a prime", *Proc. 1998 Intern'l Symp. Information Theory and Its Applications*, pp.686-689, Mexico City, Oct. 1998.

(4) T. Kaida, S. Uehara and K. Imamura, "An new algorithm for the k-error linear complexity of sequences over $GF(p^m)$ with period p^n", accepted to *the Intern'l Conference on Sequences and Their Applications* (SETA'98), Dec. 1998.