Computing the Error Linear Complexity Spectrum of a Binary Sequence of Period 2^n

Alan G.B. Lauder and Kenneth G. Paterson, *Member, IEEE*

Abstract

Binary sequences with high linear complexity are of interest in cryptography. The linear complexity should remain high even when a small number of changes are made to the sequence. The *error linear complexity spectrum* of a sequence reveals how the linear complexity of the sequence varies as an increasing number of the bits of the sequence are changed. We present an algorithm which computes the error linear complexity for binary sequences of period $\ell = 2^n$ using $O(\ell(\log \ell)^2)$ bit operations. The algorithm generalises both the Games-Chan and Stamp-Martin algorithms, which compute the linear complexity and the k-error linear complexity of a binary sequence of period $\ell = 2^n$, respectively. We also discuss an application of an extension of our algorithm to decoding a class of linear subcodes of Reed-Muller codes.

Keywords

cryptography, stream cipher, linear complexity, error linear complexity spectrum, algorithm, decoding, Reed-Muller code

I. Introduction

Stream ciphers use binary sequences with good pseudorandomness properties as key streams to encrypt messages [20]. In this context, the linear complexity of sequences is of interest. For a periodic binary sequence s this is defined to be the length of the shortest linear recurrence which generates the sequence, denoted $c(s)$. Periodic binary sequences with low linear complexity are cryptographically weak because the entire sequence can be efficiently computed given knowledge of a few initial bits using the Berlekamp-Massey algorithm [14] which requires $O(\ell^2)$ bit operations in the worst case. Here ℓ denotes the period of s. If s has period $\ell = 2^n$ then this can be improved upon by using the algorithm presented by Games and Chan in [8]. The Games-Chan algorithm computes the linear complexity of s of period $\ell = 2^n$ using $O(\ell)$ bit operations. An algorithm improving on the Games-Chan algorithm for sequences of high linear complexity can be found in [19], while algorithms which generalise both the Games-Chan algorithm and DFT-based approaches to computing linear complexity are presented in [1], [16]. These algorithms all suffer from the fact that they require as input an entire period of the sequence s to compute $c(s)$, while the Berlekamp-Massey algorithm only needs $2 \cdot c(s)$ bits. Thus they are not applicable in realistic cryptographic situations. Nevertheless, these algorithms and, more generally, the linear complexity of sequences of period 2^n, are interesting from a combinatorial perspective. For example, the papers [3], [4], [7], [9], [10] investigate the linear complexities of binary and non-binary de Bruijn sequences, while in the papers [6], [18], the linear complexities of sequences of period 2^n are a key tool in constructing sequences and arrays with certain window properties.

If a sequence has a high linear complexity, but the linear complexity can be significantly reduced by making a small number of bit changes to the sequence, then the resulting key stream is also cryptographically weak. For in this case, knowledge of the first few bits can allow the efficient generation of a sequence which closely approximates the original sequence. This observation motivates the definition of the k-error linear complexity of sequences [21] and the *stability theory* of stream ciphers [5], in which the variation of the linear complexities of sequences with the number of allowed bit changes is studied. The k-error linear complexity of a periodic binary sequence s of period ℓ is defined to be the minimum value to which the linear complexity of s can be reduced by making k or fewer bit changes in the first period of s, and identical changes in every period thereafter [21]. Thus the 0-error linear complexity of s is just $c(s)$, the linear complexity of s. We define the *error linear complexity spectrum* of s to be the ordered list of k-error linear complexities of s for $0 \leq k \leq \mathrm{wt}(s)$.

A.G.B. Lauder is a Junior Research Fellow at Wolfson College, Oxford OX2 6UD, and a member of the Mathematical Institute, Oxford University, Oxford OX1 3LB, U.K. This work was carried out while he was visiting Hewlett-Packard Laboratories, Bristol.
K.G. Paterson is with Hewlett-Packard Laboratories, Filton Road, Stoke-Gifford, Bristol BS34 8QZ, U.K.

This spectrum contains all the information about how the linear complexity of s decreases as the number k of allowed bit changes increases and is therefore a natural complexity measure to study. It is called the *k-error linear complexity profile* in [21] by analogy with the linear complexity profile introduced by Rueppel [20]. We note that Niederreiter in [17] has given an alternative definition of k-error linear complexity profile: it is defined there as a measure of how the linear complexity of s changes when considering an increasing number of initial bits of s but a fixed number k of errors.

Stamp and Martin [21] extended the Games-Chan algorithm to compute, for any fixed k, the k-error linear complexity of a binary sequence of period $\ell = 2^n$ using $O(\ell)$ arithmetic operations and $O(\ell \log \ell)$ bit operations. In contrast, no efficient algorithm is known for computing the k-error linear complexity of a sequence of general period. In [13], the first value of k for which the k-error linear complexity of s is less than $c(s)$ is shown to be an explicit function of the Hamming weight of $c(s)$ for sequences of period $\ell = 2^n$. This result reveals some information about the error linear complexity spectrum of s. Of course, the Stamp-Martin algorithm can be used in a naive fashion to compute the entire error linear complexity spectrum of a period $\ell = 2^n$ sequence using $O(\ell^2 \log \ell)$ bit operations. Our main contribution is to show, perhaps surprisingly, that an $O(\ell(\log_2 \ell)^2)$ algorithm to accomplish this task exists. This follows as an easy consequence of an algorithm we develop for a slightly more general problem. This more general algorithm also yields a soft-decoding method for a certain class of linear subcodes of Reed-Muller binary codes introduced in [15]. To our knowledge, this is the first decoding algorithm (of any type) for this class of codes.

The remainder of the paper is organised in the following way. In Section II we introduce necessary definitions and notation and present some preliminary results. Section III contains pseudocode for the main algorithm of the paper, as well as a proof of the correctness and an analysis of the computational complexity of this algorithm. Section IV contains a short example which illustrates the ideas we have employed. In Section V we discuss an extension of our algorithm which also recovers *error sequences*: these are binary sequences of weight not greater than k which indicate where bit changes should be made to reduce the linear complexity of a sequence s to the k-error linear complexity of s. We then discuss the application of this extension to the decoding of the codes of [15]. We conclude by discussing some open problems in Section VI.

II. Preliminaries

To prove the main result it is convenient to work in a more general setting. In this section we introduce the notion of a *costed binary sequence*, and in the sections which follow we present and justify an algorithm which computes the error linear complexity spectrum of such a sequence. The proof of our main result then follows easily by restricting our attention once again to ordinary binary sequences.

A. Costed binary sequences

A *costed binary sequence* S is a triple (s, σ, ℓ) where ℓ is a positive integer, $s = s[0]s[1]\ldots s[\ell-1]$ is a binary sequence of length ℓ, and $\sigma = \sigma[0]\sigma[1]\ldots\sigma[\ell-1]$ is a non-negative real sequence of length ℓ. The sequence σ is called the *cost* sequence of S and the entries of σ will record the cost of complementing bits of s. Such cost sequences are also used in [21]. Notice that we do not insist that the entries in our cost sequences should equal 1. The reasons for this are two-fold. Firstly, in order to give a rigorous and compact proof of the correctness of our algorithm, we will need a more general notion than just the *number* of bits when accounting for the cost of complementing bits in sequences. Secondly, in the application to decoding, we will want to consider real-valued costs.

In what follows, we shall only consider costed binary sequences whose length is a power of 2.

We define two maps on the set of all costed binary sequences whose length is a positive power of 2. Let $S = (s, \sigma, \ell)$ be such a sequence. The first map B is defined as follows: Let $B(S) = (B(s), B(\sigma), \ell/2)$ where $B(s)$ and $B(\sigma)$ are given by the pseudo-code in the following routine, c.f. [21, Fig. 3]:

`INPUT:` $S = (s, \sigma, \ell)$
`OUTPUT:` $B(S) = (B(s), B(\sigma), \ell/2)$

`for` $0 \leq i < \ell/2$

$B(s)[i] = s[i] \oplus s[i + (\ell/2)]$
$B(\sigma)[i] = \min\{\sigma[i], \sigma[i + (\ell/2)]\}$

The second map L is defined as follows: Let $L(S) = (L(s), L(\sigma), \ell/2)$ where $L(s)$ and $L(\sigma)$ are given by the pseudo-code in the following routine, also c.f. [21, Fig. 3]:

```
INPUT:  S = (s, σ, ℓ)
OUTPUT: L(S) = (L(s), L(σ), ℓ/2)

 for 0 ≤ i < ℓ/2
    if s[i] = s[i + (ℓ/2)] then
        L(s)[i] = s[i]
        L(σ)[i] = σ[i] + σ[i + (ℓ/2)]
    if s[i] ≠ s[i + (ℓ/2)] then
        if σ[i] > σ[i + (ℓ/2)] then
            L(s)[i] = s[i]
            L(σ)[i] = σ[i] − σ[i + (ℓ/2)]
        else
            L(s)[i] = s[i + (ℓ/2)]
            L(σ)[i] = σ[i + (ℓ/2)] − σ[i]
```

We also define similar maps B and L for binary sequences. Given a binary sequence s of length ℓ a positive power of 2, we define $B(s)$ by

$$B(s)[i] = s[i] \oplus s[i + (\ell/2)], \quad 0 \leq i < \ell/2.$$

If s is a binary sequence of length ℓ such that $B(s) = 0$ then we define $L(s)$ by

$$L(s)[i] = s[i] \;\; (= s[i + (\ell/2)]), \quad 0 \leq i < \ell/2.$$

Observe that the definitions given for sequences are consistent with those given for costed binary sequences.

B. Linear complexity

Given a non-zero, infinite periodic binary sequence $s^\infty = s[0]s[1]\ldots$, the *linear complexity* of s^∞ is defined to be the smallest c such that there exists binary constants $a_0, a_1, \ldots, a_c$, not all zero, with $\sum_{0 \leq i \leq c} a_i s[j + i] = 0$ for all $j \geq 0$. We define the linear complexity of the zero sequence to be 0. Let ℓ denote the period of s^∞. We identify s^∞ with the finite, length ℓ binary sequence $s = s[0]s[1]\ldots s[\ell - 1]$ and denote the linear complexity of s^∞ by $c(s)$.

We will need a key lemma, proved as [8, Theorem 6].

Lemma 1: Let s be a binary sequence of length 2^n. Then

1. $c(s) = 2^{n-1} + c(B(s))$ if $B(s) \neq 0$
2. $c(s) = c(L(s))$ if $B(s) = 0$.

This lemma shows how the computation of the linear complexity of a sequence of period 2^n can be efficiently reduced to the same computation for a sequence of period only 2^{n-1}.

C. Error sequences and k-error linear complexity

Let $S = (s, \sigma, \ell)$ be a costed binary sequence. An *error sequence* for S is just a binary sequence $e = e[0]e[1]\ldots e[\ell - 1]$ of length l. We define $\mathrm{cost}(s \to s \oplus e) = \sum_{e[i]=1} \sigma[i]$. Thus $\mathrm{cost}(s \to s \oplus e)$ represents the total cost of changing s by the error sequence e if the cost of changing bit $s[i]$ is $\sigma[i]$.

For k a non-negative real number, the *k-error linear complexity* of S, denoted $c_k(S)$, is defined to be

$$\min_{\mathrm{cost}(s \to s \oplus e) \leq k} c(s \oplus e)$$

where the min is taken over all error sequences for S whose "costs" are no more than k. So $c_0(S) = c(s)$ if all the entries in σ are positive. Notice also that if $\sigma[i] = 1$ for each i, then the definition of k-error linear complexity given here agrees with that given in Section I.

The following lemma is central to our paper and shows how error sequences with certain costs for $B(S)$ and $L(S)$ can be used to devise error sequences for S with related costs.

Lemma 2: Let $S = (s, \sigma, \ell)$ be a costed binary sequence with $\mathrm{cost}(B(s) \to 0) = T$ (this cost is calculated using $B(\sigma)$).

1. (a) If f is an error sequence for $B(S)$ then there exists an error sequence e for S such that $B(s \oplus e) = B(s) \oplus f$ and $\mathrm{cost}(B(s) \to B(s) \oplus f) = \mathrm{cost}(s \to s \oplus e)$.

(b) If e is an error sequence for S then there exists an error sequence f for $B(S)$ such that $B(s \oplus e) = B(s) \oplus f$ and $\mathrm{cost}(B(s) \to B(s) \oplus f) \leq \mathrm{cost}(s \to s \oplus e)$.

In these two cases, $B(\sigma)$ is used to calculate $\mathrm{cost}(B(s) \to B(s) \oplus f)$.

2. (a) If f is an error sequence for $L(S)$ then there exists an error sequence e for S such that $L(s \oplus e) = L(s) \oplus f$ with $B(s \oplus e) = 0$ and $\mathrm{cost}(L(s) \to L(s) \oplus f) + T = \mathrm{cost}(s \to s \oplus e)$.

(b) If e is an error sequence for S with $B(s \oplus e) = 0$ then there exists an error sequence f for $L(S)$ such that $L(s \oplus e) = L(s) \oplus f$ and $\mathrm{cost}(L(s) \to L(s) \oplus f) + T = \mathrm{cost}(s \to s \oplus e)$.

In these two cases, $L(\sigma)$ is used to calculate $\mathrm{cost}(L(s) \to L(s) \oplus f)$.

Proof: In each case we exhibit an error sequence which can be verified to be of the correct form. We omit the relatively routine calculations involved in this checking.

1. (a) Let $f = f[0]f[1] \dots f[(\ell/2) - 1]$ be an error sequence for $B(S)$. Define the error sequence $e = e[0]e[1] \dots e[\ell - 1]$, which we call the B*-pull-up* of f, as follows:

```
INPUT:  S = (s, σ, ℓ), f
OUTPUT: e, the B-pull-up of f
  for 0 ≤ i < ℓ/2
    if f[i] = 1 then
        if σ[i] ≤ σ[i + (ℓ/2)] then
            e[i] = 1 and e[i + (ℓ/2)] = 0
        else
            e[i] = 0 and e[i + (ℓ/2)] = 1
    else
        e[i] = e[i + (ℓ/2)] = 0
```

(b) Let e be an error sequence for S. Define $f = B(e)$. (Observe that f is the unique error sequence whose B-pull-up is e).

2. (a) Let f be an error sequence for $L(S)$. Define the error sequence $e = e[0]e[1] \dots e[\ell - 1]$ for S, which we call the L*-pull-up* of f, in the following way.

```
INPUT:  S = (s, σ, ℓ), f
OUTPUT: e, the L-pull-up of f
  for 0 ≤ i < ℓ/2
    if B(s)[i] = 0 then
        e[i] = e[i + (ℓ/2)] = f[i]
    else \* Case B(s)[i] = 1 *\
        if σ[i] ≤ σ[i + (ℓ/2)] then
            e[i] = f[i] ⊕ 1 and e[i + (ℓ/2)] = f[i]
        else
            e[i] = f[i] and e[i + (ℓ/2)] = f[i] ⊕ 1
```

(b) Let e be an error sequence for S such that $B(s \oplus e) = 0$. Observe here that $B(e) = B(s)$. We take f to be the unique error sequence whose L-pull-up is e. The sequence f may be explicitly defined as follows:

```
INPUT:  S = (s, σ, ℓ), e
OUTPUT: f
  for 0 ≤ i < ℓ/2
    if B(e)[i] = 0 then \* Case B(e)[i] = B(s)[i] = 0 *\
        f[i] = e[i]
    else \* Case B(e)[i] = B(s)[i] = 1 *\
        if σ[i] ≤ σ[i + (ℓ/2)] then
            f[i] = e[i + (ℓ/2)]
        else
            f[i] = e[i]
```

■

We now present an analogue of Lemma 1 showing how the computation of the k-error linear complexity of the period 2^n sequence S can be reduced to the computation of either the k-error linear complexity of $B(S)$ or the $(k-T)$-error linear complexity of $L(S)$, these being sequences of period 2^{n-1}. The lemma can be proved from the assumption that the algorithm of Stamp and Martin [21] is correct. However, we prefer to give an independent and completely rigorous proof based on Lemma 2.

Lemma 3: Let $S = (s, \sigma, 2^n)$ be a costed binary sequence and $k \geq 0$. Let $T = \mathrm{cost}(B(s) \to 0)$. Then $c_k(S)$ is equal to
1. $2^{n-1} + c_k(B(S))$ in the case that $0 \leq k < T$.
2. $c_{k-T}(L(S))$ in the case that $T \leq k$.

Proof: We use Lemma 2 to prove Part 2; Part 1 may be proved in a similar way (using B-pull-ups instead of L-pull-ups).

We first show that $c_k(S) \leq c_{k-T}(L(S))$. Suppose that we have an error sequence f for $L(S)$ which minimises the linear complexity of $L(S)$ among all error sequences whose "costs" are not greater than $k - T$. Hence $c(L(s) \oplus f) = c_{k-T}(L(S))$ and $\mathrm{cost}(L(s) \to L(s) \oplus f) \leq k - T$. Using Part 2(a) of Lemma 2, we take the L-pull-up of f to obtain an error sequence e for S such that $L(s \oplus e) = L(s) \oplus f$, $B(s \oplus e) = 0$ and

$$\mathrm{cost}(s \to s \oplus e) = T + \mathrm{cost}(L(s) \to L(s) \oplus f) \leq T + (k - T) = k.$$

By Part 2 of Lemma 1, $c(s \oplus e) = c(L(s \oplus e)) = c(L(s) \oplus f) = c_{k-T}(L(S))$. Hence e is an error vector for S with $\mathrm{cost}(s \to s \oplus e) \leq k$ and $c(s \oplus e) = c_{k-T}(L(S))$. It follows that $c_k(S) \leq c_{k-T}(L(S))$.

We now show that $c_{k-T}(L(S)) \leq c_k(S)$. Observe first that if e is an error sequence for S which minimises the linear complexity of S over all error sequences whose "costs" are not greater than $k \geq T$, then we must have that $B(s \oplus e) = 0$. For suppose that $B(s \oplus e) \neq 0$. Then taking e' to be the B-pull-up of $B(s)$ we find that $B(s \oplus e') = 0$ and $\mathrm{cost}(s \to s \oplus e') = T \leq k$. So by Part 2 of Lemma 1, $c(s \oplus e') = c(L(S \oplus e')) \leq 2^{n-1}$. But since $B(s \oplus e) \neq 0$ we have $c(s \oplus e) > 2^{n-1}$ by Part 1 of Lemma 1, which gives us a contradiction.

Hence $B(s \oplus e) = 0$ and we may apply part 2(b) of Lemma 2 to deduce that there exists an error sequence f for $L(S)$ such that $L(s \oplus e) = L(s) \oplus f$ and $\mathrm{cost}(s \to s \oplus e) = T + \mathrm{cost}(L(s) \to L(s) \oplus f)$. Then

$$\mathrm{cost}(L(s) \to L(s) \oplus f) = \mathrm{cost}(s \to s \oplus e) - T \leq k - T.$$

Since $B(s \oplus e) = 0$, Part 2 of Lemma 1 applies to show that $c(L(s) \oplus f) = c(L(s \oplus e)) = c(s \oplus e) = c_k(s)$. Hence f is an error vector for $L(S)$ with $\mathrm{cost}(L(s) \to L(s) \oplus f) \leq k - T$ and $c(L(s) \oplus f) = c_k(s)$. It follows that $c_{k-T}(L(S)) \leq c_k(S)$.

This completes the proof of Part 2. ■

D. Error Linear Complexity Spectrum

We define the *error linear complexity spectrum (ELCS)* of $S = (s, \sigma, 2^n)$ to be the set of pairs

$$\{(k, c_k(S)) : 0 \leq k \leq \mathrm{cost}(s \to 0)\}.$$

Notice here that we do not restrict k to be an integer, so the ELCS is not a finite set. Of course, the ELCS of S can be visualised as a graph with axes for cost and linear complexity and with points $(k, c_k(S))$,

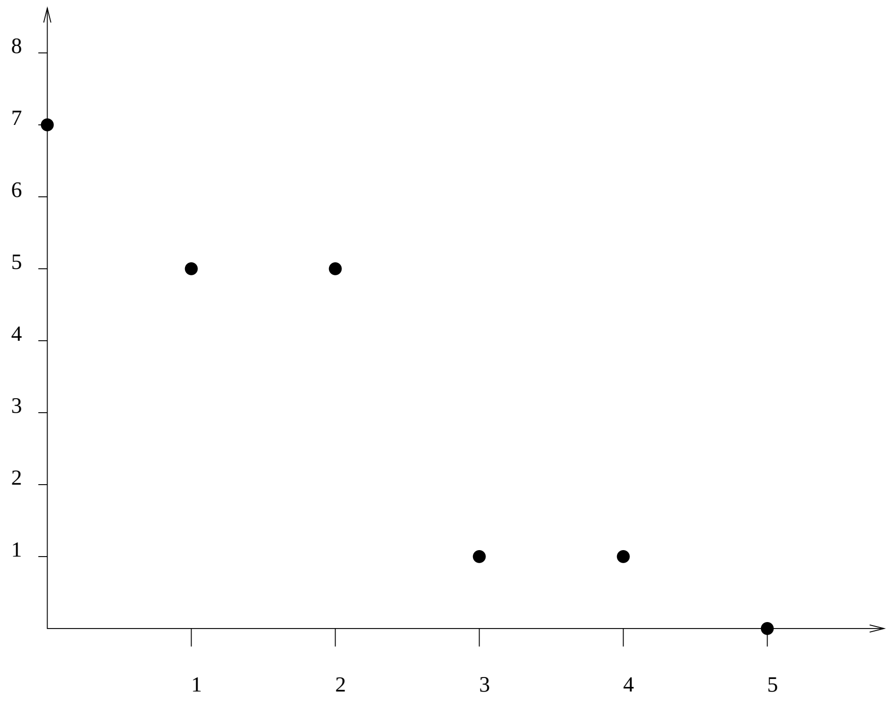

Fig. 1. The graph of the ELCS of sequence $S = (s, \sigma, 8)$ with $s = 10011110$ and $\sigma = 11111111$. Notice how the graph is determined by the set of critical points $\{(0, 8), (1, 5), (3, 1), (5, 0)\}$.

$0 \leq k \leq \mathrm{cost}(s \to 0)$. For example, we will see in Section IV that the costed binary sequence $S = (s, \sigma, 8)$ with

$$s = 10011110 \quad \text{and} \quad \sigma = 11111111$$

has ELCS as shown in Figure 1.

The following lemma shows how the ELCS of $B(S)$ and of $L(S)$ can be joined together to give the ELCS of S.

Lemma 4: Let $S = (s, \sigma, 2^n)$ be a costed binary sequence. Let $B(S) = (B(s), B(\sigma), 2^{n-1})$ with $T = \mathrm{cost}(B(s) \to 0)$, and $L(S) = (L(s), L(\sigma), 2^{n-1})$ with $U = \mathrm{cost}(L(s) \to 0)$. Suppose that the ELCS of $B(S)$ is $\{(k, c_k(B(S))) : 0 \leq k \leq T\}$ and the ELCS of $L(S)$ is $\{(k, c_k(L(S))) : 0 \leq k \leq U\}$. Then the ELCS of S is

$$\{(k, c_k(B(S)) + 2^{n-1}) : 0 \leq k < T\} \cup \{(T + k, c_k(L(S))) : 0 \leq k \leq U\}$$

Proof: Follows immediately from Lemma 3. ■

E. Critical Error Linear Complexity Spectrum

Let $(k, c_k(S))$ be a point on the ELCS of the costed binary sequence S. We say that $(k, c_k(S))$ is *critical* if for all points $(k', c_{k'}(S))$ of the ELCS with $k' < k$ we have that $c_{k'}(S) > c_k(S)$. In other words, critical points are the points on the graph of the ELCS where a decrease in the k-error linear complexity occurs. The sublist of all critical points in the ELCS of S is called the *critical error linear complexity spectrum (CELCS)* of S. We include the point $(0, c(s))$ in the CELCS of S. Thus the main result of [13] explicitly gives the k value of the second critical point of a sequence of period 2^n. Because the linear complexity of $s \oplus e$ can take on only finitely many different values, the CELCS of S contains a finite number of points. Observe that the CELCS of a costed binary sequence entirely determines its ELCS and vice versa, hence the terminology *critical.* For example, the CELCS of the period 8 sequence S defined above is the set:

$$\{(0,8),(1,5),(3,1),(5,0)\}.$$

Lemma 5: Let $S=(s,\sigma,2^n)$ be a costed binary sequence. Let $B(S)=(B(s),B(\sigma),2^{n-1})$ with $\mathrm{cost}(B(s)\to 0)=T$, and $L(S)=(L(s),L(\sigma),2^{n-1})$ with $\mathrm{cost}(L(s)\to 0)=U$. Let the CELCS of $B(S)$ be $\{(k_i,c_{k_i}(B(S))) : 0\le i\le t\}$ for some t, where $k_0=0$ and $k_t=T$. Furthermore, let the CELCS of $L(S)$ be $\{(K_i,c_{K_i}(L(S))) : 0\le i\le u\}$ for some u where $K_0=0$ and $K_u=U$. Then the CELCS of S is

$$\{(k_0,c_{k_0}(B(S))+2^{n-1}),(k_1,c_{k_1}(B(S))+2^{n-1}),\dots,(k_{t-1},c_{k_{t-1}}(B(S))+2^{n-1}),$$
$$(T+K_0,c_{K_0}(L(S))),(T+K_1,c_{K_1}(L(S))),\dots(T+K_u,c_{K_u}(L(S)))\}$$

Proof: Follows from Lemma 4. ∎

III. An Algorithm to Compute the CELCS of a Costed Binary Sequence

We now present our algorithm for computing the critical error linear complexity spectrum of a costed binary sequence of period $\ell=2^n$. As well as a costed binary sequence $S=(s,\sigma,\ell)$, the algorithm has as inputs three integers `tsf`, `lim` and c, whose function will be explained shortly.

Algorithm `CELCS(`$S=(s,\sigma,\ell)$`,tsf,lim,`c`)`

```
if l > 1
    calculate B(S) and L(S)
    let T = cost(B(s) → 0)
    if T > 0
        CELCS((B(s), B(σ), ℓ/2), tsf, min{lim, tsf + T − 1}, c + (ℓ/2))
    if tsf + T ≤ lim
        CELCS((L(s), L(σ), ℓ/2), tsf + T, lim, c)
else \* Case ℓ = 1 *\
    if s[0] = 0
        output (tsf, c)
    if s[0] = 1 and σ[0] > 0
        output (tsf, c + 1)
    if s[0] = 1 and tsf + σ[0] ≤ lim
        output (tsf + σ[0], c)
```

A. Notes on the algorithm

The algorithm is recursive, calling itself with progressively shorter costed sequences as input. It can be thought of as exploring a binary tree where a node at depth $i\,(0\le i\le n)$ corresponds to a costed binary sequence of length 2^{n-i} and the two edges emanating from a node correspond to the two mappings B and L that can be applied to this sequence. At any stage in the execution of the algorithm, the variable `tsf` is set to the *total* cost of changes made to the sequence *so far*, and the variable `lim` marks a *limit* to the total cost of changes one should consider when searching a particular part of the tree. We will establish below that execution of the algorithm `CELCS`$((s,\sigma,2^n),0,N,0)$ where $N=\mathrm{cost}(s\to 0)$ will produce as output the CELCS of the costed binary sequence $(s,\sigma,2^n)$.

B. Correctness of Algorithm `CELCS`

We require one preliminary lemma from which our main results follow very quickly.

Lemma 6: The algorithm `CELCS`$((s,\sigma,\ell),\mathtt{tsf},\mathtt{lim},c)$, where $\mathtt{tsf}\le\mathtt{lim}$, outputs the following list of points:

$$\{(\mathtt{tsf}+k_0,c+c_{k_0}(S)),(\mathtt{tsf}+k_1,c+c_{k_1}(S)),\dots(\mathtt{tsf}+k_u,c+c_{k_u}(S))\}$$

where $(k_0, c_{k_u}(S)), \ldots, (k_u, c_{k_u}(S))$ are the critical points of $S = (s, \sigma, \ell)$ whose first coordinates lie in the range $[0, \texttt{lim} - \texttt{tsf}]$.

Proof: We prove the lemma by induction on ℓ. If $l = 1$ then there are two cases to consider.
1. $\mathrm{cost}(s \to 0) = 0$. In this case the single critical point whose first coordinate lies between 0 and $\texttt{lim} - \texttt{tsf}$ is $(0, 0)$. The algorithm outputs $\{(\texttt{tsf}, c)\}$ as required.
2. $\mathrm{cost}(s \to 0) > 0$. Here $s[0] = 1$ and $\sigma[0] > 0$. If $\sigma[0] \leq \texttt{lim} - \texttt{tsf}$ then the two critical points with first coordinate in the range 0 to $\texttt{lim} - \texttt{tsf}$ are $(0, 1)$ and $(\sigma[0], 0)$. The algorithm outputs

$$\{(\texttt{tsf}, c+1), (\texttt{tsf} + \sigma[0], c)\}$$

as required. If $\sigma[0] > \texttt{lim} - \texttt{tsf}$ then the single critical point with first coordinate in the range 0 to $\texttt{lim} - \texttt{tsf}$ is $(0, 1)$. Once again, the algorithm correctly outputs $(\texttt{tsf}, c+1)$. This completes the consideration of length 1 sequences.

Assume now that the result is true for all costed binary sequences of length $2^{n'}$ with $n' < n$. Let $S = (s, \sigma, \ell)$ be a costed binary sequence of length $\ell = 2^n > 1$.

The algorithm $\mathrm{CELCS}((s, \sigma, \ell), \texttt{tsf}, \texttt{lim}, c)$, where $\texttt{tsf} \leq \texttt{lim}$, calls as subroutines the two algorithms
1. $\mathrm{CELCS}((B(s), B(\sigma), \ell/2), \texttt{tsf}, \min(\texttt{tsf} + T - 1, \texttt{lim}), c + (\ell/2))$, where $\mathrm{cost}(B(s) \to 0) = T$. Observe here that $\texttt{tsf} \leq \min(\texttt{tsf} + T - 1, \texttt{lim})$ because $T > 0$.
2. $\mathrm{CELCS}((L(s), L(\sigma), \ell/2), \texttt{tsf} + T, \texttt{lim}, c)$, where $\mathrm{cost}(B(s) \to 0) = T$. (This call is only made when $\texttt{tsf} + T \leq \texttt{lim}$, and so in particular $\min(\texttt{tsf} + T - 1, \texttt{lim}) = \texttt{tsf} + T - 1$.)

By the inductive hypothesis, the first of these outputs

$$\{(\texttt{tsf} + k_0, (c + (\ell/2)) + c_{k_0}(B(S))), \ldots (\texttt{tsf} + k_v, (c + (\ell/2)) + c_{k_v}(B(S)))\}$$

where $(k_0, c_{k_0}(B(S))), \ldots, (k_v, c_{k_v}(B(S)))$ are the critical points of $B(S)$ whose first coordinates lie in the range $[0, \min(T - 1, \texttt{lim} - \texttt{tsf})]$. (Observe that $\min(T - 1, \texttt{lim} - \texttt{tsf}) = \min(\texttt{tsf} + T - 1, \texttt{lim}) - \texttt{tsf}$.) By Lemma 5, if $\texttt{lim} - \texttt{tsf} \leq T - 1$ then this is the whole of the required output. Similarly by Lemma 5, if $\texttt{lim} - \texttt{tsf} \geq T$ then this is the first portion of the required output, the remainder coming from the second recursive call which we now describe.

¿From the inductive hypothesis we know that the second subroutine in the case $\texttt{lim} - \texttt{tsf} \geq T$ will output

$$\{((\texttt{tsf} + T) + K_0, c + c_{K_0}(L(S))), \ldots ((\texttt{tsf} + T) + K_u, c + c_{K_u}(L(S)))\}$$

where $(K_0, c_{K_0}(L(S))), \ldots (K_u, c_{K_u}(L(S)))$ are the critical points of $L(S)$ whose first coordinates lie in the range $[0, \texttt{lim} - (\texttt{tsf} + T)]$. By Lemma 5 this is the second portion of the required output.

Thus the algorithm outputs the claimed list for costed binary sequences of length 2^n. This completes the proof. ■

Theorem 7: Let $S(s, \sigma, 2^n)$ be a costed binary sequence and write $N = \mathrm{cost}(s \to 0)$. Then the algorithm $\texttt{CELCS}((s, \sigma, 2^n), 0, N, 0)$ correctly outputs the critical error linear complexity spectrum of S.

Proof: By Lemma 6, with this input, the algorithm outputs the critical points of $S = (s, \sigma, \ell)$ whose first coordinates lie between 0 and N. This is the complete critical error linear complexity spectrum of S because $\mathrm{cost}(s \to 0) = N$ implies that the critical point with highest first coordinate is $(N, 0)$. ■

Now we revert to considering the error linear complexity spectra of binary sequences, as defined in Section I. Let s^∞ be a binary sequence of period $\ell = 2^n$. If we associate with s^∞ the costed binary sequence $S = (s, \sigma, 2^n)$ where s is the finite sequence $s[0]s[1] \ldots s[\ell - 1]$ and $\sigma[i] = 1$ for $0 \leq i < 2^n$, then it is easy to see that the error linear complexity spectrum of s^∞, as defined in Section I, is just the error linear complexity spectrum of S. This is because allowing up to k bit changes in each period of s^∞ is equivalent, with this choice of σ, to considering error sequences e satisfying $\mathrm{cost}(s \to s \oplus e) \leq k$.

So knowledge of the critical error linear complexity spectrum of S will entirely determine the error linear complexity spectrum of s^∞. Thus by Theorem 7, the algorithm $\texttt{CELCS}(S, 0, 2^n, 0)$ can be used to determine the error linear complexity spectrum of s^∞.

C. *Performance of Algorithm* CELCS

We now consider the computational complexity of our algorithm. For simplicity, we assume that the entries in the cost vectors are scaled and quantised to be non-negative integers rather than real numbers.

Theorem 8: Let $S = (s, \sigma, 2^n)$ be a costed binary sequence and suppose $N = \text{cost}(s \to 0)$. Then the algorithm $\texttt{CELCS}(S, 0, N, 0)$ outputs the complete critical error linear complexity spectrum of S in $O(2^n n(n + \log_2 M))$ bit operations, where each entry in the cost sequence σ is an integer no greater than M.

Proof: The algorithm explores part of a binary tree with each node at depth i having a computational cost of $O(2^{n-i}(\log_2 M + i))$ bit operations. Here, we have taken into account the possible doubling in size of the components of the cost sequences at each depth in the tree. Hence the total computational cost is

$$O\left(\sum_{0 \le i \le n} 2^i 2^{n-i}(\log_2 M + i)\right) = O\left(2^n n(n + \log_2 M)\right)$$

bit operations. ■

Taking $M = 1$ in the above proof, we get:

Corollary 9: Algorithm CELCS can correctly output the complete critical error linear complexity spectrum of a binary sequence of period $\ell = 2^n$ in $O(\ell(\log_2 \ell)^2)$ bit operations.

Observe that the computational complexity of the Stamp-Martin algorithm in [21, Figure 3] is $O(\ell \log_2 \ell)$ bit operations. Stamp and Martin give a running time of "$O(\ell)$ steps". This actually measures the arithmetic complexity of the Stamp-Martin algorithm, as they did not consider the growth of the costs. Thus our algorithm finds the whole spectrum at the expense of only an additional factor of $\log_2 \ell$ in the number of bit operations.

IV. An Example

Consider the periodic binary sequence s^∞ obtained by repetition of the finite sequence $s = 10011110$ introduced in Section II. One may associate with s^∞ the costed binary sequence $S = (s, \sigma, 8)$ where the cost sequence σ consists solely of 1's. The execution of the algorithm $\texttt{CELCS}(S, 0, 8, 0)$ is depicted in Figure 2. We have used the notation $s[i]_{\sigma[i]}$ to record sequence elements and their costs. The figure reveals that the critical error linear complexity spectrum of S is

$$\{(0, 8), (1, 5), (3, 1), (5, 0)\}.$$

Observe that the binary sequence s has odd weight 5, and recall that all binary sequences of period a power of 2 with odd weight have maximal linear complexity. Thus to reduce the linear complexity of S one must make an *odd* number of changes. This explains the fact that the first coordinates of the critical error points $(k, c_k(S))$, for $k > 0$, all have equal parity. This will always be true for the CELCS of binary sequences of period a power of 2 when the cost sequence is the all 1's sequence.

V. Error sequences and an application to coding theory

A. *Critical error sequences*

Let $(k, c_k(S))$ be a point on the CELCS of the costed binary sequence $S = (s, \sigma, \ell)$. Let e be an error sequence for S such that $c(s \oplus e) = c_k(S)$ and $\text{cost}(s \to s \oplus e) = k$. We call e a *critical error sequence* for S. An ordered list of critical error sequences for each critical point $(k, c_k(S))$ is called a *critical error list.* Observe that such a list is not necessarily unique. (More generally, we can consider error sequences for each point on the ELCS of S, but from our definitions, it should be clear that such error sequences can be taken to be critical ones.)

Let $S = (s, \sigma, 2^n)$ and f be a critical error sequence for $B(S)$. Then it follows from Lemma 2 and the proof of Lemma 3 that the B-pull-up of f is a critical error sequence for S. Similarly, given a critical error sequence f for $L(S)$ one may take the L-pull-up and obtain a critical error sequence for S. It then follows easily that

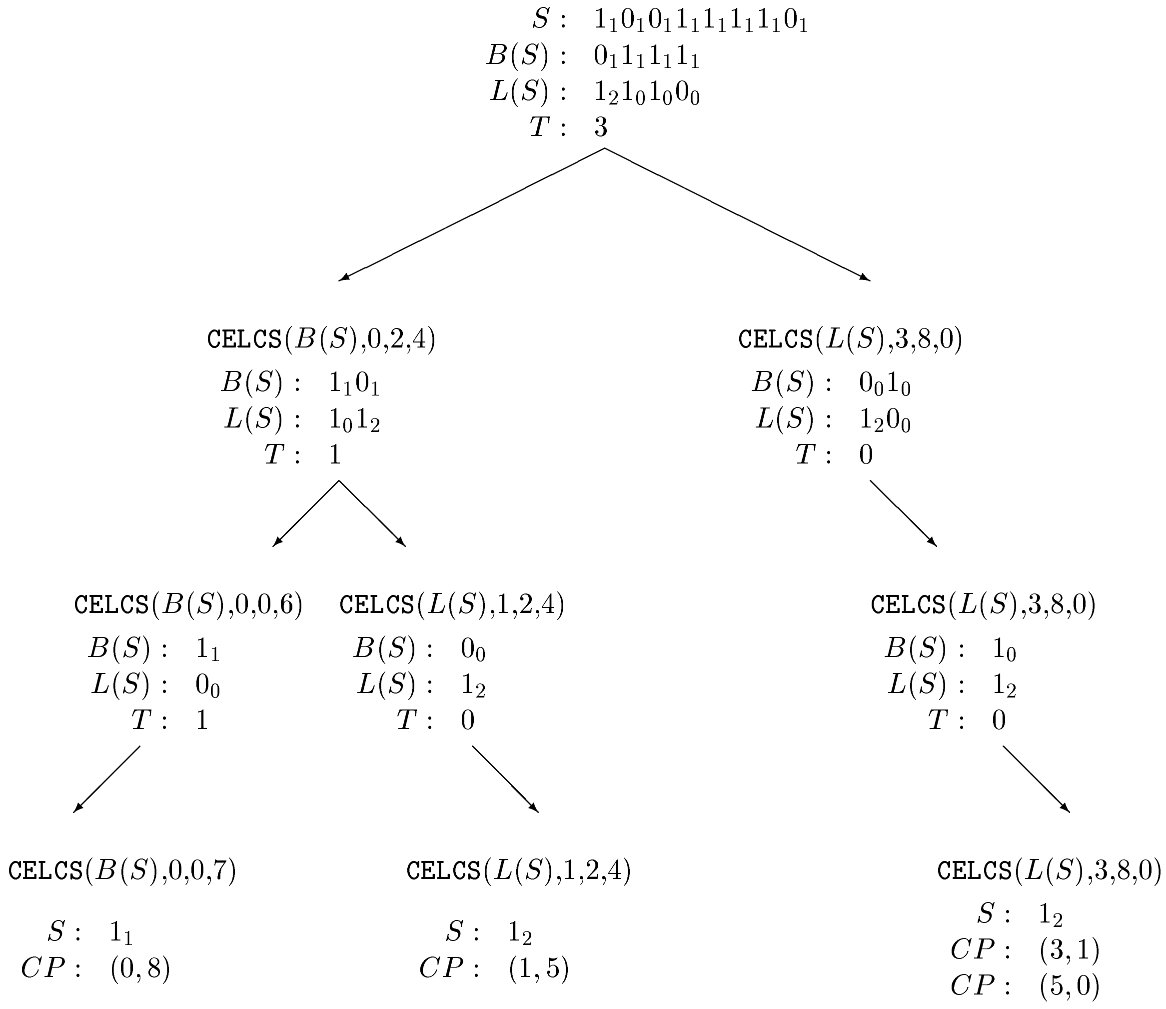

Fig. 2. Schematic execution of Algorithm CELCS$(S, 0, 8, 0)$

if we take the union of the B-pull-up of a critical error list of $B(S)$ and the L-pull-up of a critical error list for $L(S)$, we will obtain a critical error list for S itself. We also note that it is trivial to write down critical error sequences for the three cases for $\ell = 1$ that occur in Algorithm CELCS:

```
INPUT:  S = (s, σ, 1) of length 1, tsf, lim
OUTPUT: The critical error list for critical points of S with first coordinate
in the range [0, lim − tsf]

    if s[0] = 0
        output e = 0
    if s[0] = 1 and σ[0] > 0
        output e = 0
    if s[0] = 1 and tsf + σ[0] ≤ lim
        output e = 1
```

Thus one may use the subroutines presented in the proof of Lemma 2 for computing B- and L-pull-ups to devise an algorithm very similar to Algorithm CELCS which computes a critical error list for a costed binary sequence. It is also possible to compute single critical error sequences by reversing the Stamp-Martin algorithm and using B- and L-pull-ups. For an alternative approach to finding critical error sequences, see [11].

Example 10: We re-visit the costed binary sequence S of Section IV. Recall that $(3, 1)$ is a point on the CELCS of s obtained, according to Figure 2, by applying L three times to s and then applying the second of the three cases for $\ell = 1$. Reversing these steps, we begin with $e = 0$ and take three L-pull-ups (using the appropriate cost vector at each stage), obtaining the sequence of critical error sequences:

$$0 \to 01 \to 0001 \to 01100001.$$

Thus $e = 01100001$ is a critical error sequence for S at the point $(3, 1)$. Indeed $s \oplus e = 11111111$ has linear complexity 1 and $\mathrm{cost}(s \to s \oplus e) = 3$.

B. A decoding algorithm for some subcodes of the Reed-Muller codes

For any $2 \le u \le n$ we define the code $\mathcal{C}(n, u)$ to be the vector space of all binary sequences s of length 2^n whose linear complexity $c(s)$ is at most $2^{n-u+1} - 1$. In [15, Theorem 2] it is shown that $\mathcal{C}(n, u)$ is a $[2^n, 2^{n-u+1} - 1, 2^u]$ binary, linear code. In fact, it is a subcode of the $(n-u)$-th order Reed-Muller code with the same minimum distance.

Given an arbitrary received sequence r (which we assume to be a binary vector of length 2^n), to find the closest codeword $c \in \mathcal{C}(n, u)$ to r (in terms of Hamming distance), we must find the minimum value of k such that the linear complexity of r is reduced to less than or equal $2^{n-u+1} - 1$ by making exactly k changes to r. We can use the ideas presented in the preceding sections to solve this problem. Firstly, we compute the CELCS of the costed binary sequence $R = (r, \sigma, 2^n)$ where $\sigma = 11\ldots1$, by executing the algorithm $\mathtt{CELCS}(R, 0, 2^n, 0)$. Next we choose a point $(k, c_k(R))$ in the spectrum of R with k minimal subject to the condition that $c_k(R) \le 2^{n-u+1} - 1$. We assume that this point is a critical point without any loss. Finally, we recover a critical error sequence e for the point $(k, c_k(R))$ using the method of Section V-A. The sequence e is an error vector of minimum weight for r and the codeword closest to r is $c = r \oplus e$. The computational complexity of this minimum distance decoding algorithm is $O(\ell(\log \ell)^2)$ bit operations, where $\ell = 2^n$ is the code length.

Example 11: We consider the code $C(3, 2)$ consisting of all length 8 sequences with linear complexity at most 3 and a received word $r = 10011110$. We showed in the example of Section IV that the CELCS of r is:

$$\{(0, 8), (1, 5), (3, 1), (5, 0)\}.$$

Thus the minimum number of changes that we can make to r to reduce its linear complexity to 3 or lower is 3 (in fact $(3, 1)$ is a point on the CELCS of r, so this number of changes can reduce the linear complexity to just 1). We showed in Example 10 that a critical error sequence for the point $(3, 1)$ is $e = 01100001$. Hence the nearest codeword to r in $\mathcal{C}(3, 2)$ is $c = r \oplus e = 11111111$.

It has been observed [2] that by allowing arbitrary cost sequences σ in this decoding algorithm, we can obtain a soft decision decoding algorithm for the code $\mathcal{C}(m, u)$. The input to the algorithm is now a costed binary sequence $R = (r, \sigma, 2^n)$ in which r is a binary, hard-decision version of the received vector and σ is a real vector recording the unreliability of these hard decisions.

VI. Conclusion

We have exhibited an algorithm which computes the error linear complexity spectrum of a binary sequence of period $\ell = 2^n$ using $O(\ell(\log \ell)^2)$ bit operations. Furthermore, we have described how this algorithm may be adapted to output a critical error list which specifies exactly which changes must be made to reduce the linear complexity of the sequence by increasing amounts. This modified algorithm may be used to give a soft-decision decoding algorithm for a particular class of subcodes of Reed-Muller codes.

The formulation of our algorithm as a partial exploration of a binary tree with branching depending on explicitly computed quantities may lend it to further analysis. For example, it would be of interest to investigate the distribution of error linear complexity spectra for random sequences of a given period. Related information, such as the average number of critical points, might be easier to obtain, while characterising the non-zero sequences with the maximum or minimum number of critical points also seems feasible.

We note that reasonably efficient algorithms already exist to compute the linear complexities and the k-error linear complexities of sequences of period p^n over finite fields of characteristic p [5], [12]. It seems plausible that these can be adapted to compute error linear complexity spectra too. But it remains a challenging open problem to devise an algorithm which efficiently computes the error linear complexity spectra of binary sequences of arbitrary period.

References

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

[2] S.R. Blackburn, *personal communication*.

[3] S.R. Blackburn, T. Etzion and K.G. Paterson, "Permutation Polynomials, de Bruijn Sequences and Linear Complexity", *Journal of Comb. Theory Ser. A*, **76**, pp. 55-82, 1996.

[4] A.H. Chan, R.A. Games and E.L. Key, "On the complexities of de Bruijn sequences", *J. Comb. Theory, Ser. A*, **33**, pp. 233-246, 1982.

[5] C. Ding, G, Xiao and W. Shan, *The stability theory of stream ciphers*, Lecture Notes in Computer Science, Vol. 561, Springer-Verlag, Berlin, 1991.

[6] T. Etzion, "Constructions for perfect maps and pseudo-random arrays", *IEEE Trans. Inform. Theory*, **IT-34**, pp. 1308-1316, 1988.

[7] T. Etzion and A. Lempel, "Construction of de Bruijn sequences of minimal complexity. *IEEE Trans. Inform. Theory*, **IT-30**, pp. 705-709, 1984.

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

[9] P.A. Hines, "Characterising the linear complexity of span 1 de Bruijn sequences over finite fields", *J. Combin. Theory Ser. A* **81**(2), pp. 140–148, 1998.

[10] P.A. Hines, "On the minimum linear complexity of de Bruijn sequences over non-prime finite fields", *J. Combin. Theory Ser. A* **86**(1), pp. 127–139, 1999.

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

[12] 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", *Information and Computation*, **151**, pp. 134-147, 1999.

[13] K. Kurosawa, F. Sato, T. Sakata and W. Kishimoto,"A relationship between linear complexity and k-error linear complexity", *IEEE Trans. Inform. Theory*, **IT-46**, pp. 694-698, 2000.

[14] J.L. Massey, "Shift Register Synthesis and BCH decoding", *IEEE Trans. Inform. Theory*, **IT-15**, pp. 122-127, Jan. 1969.

[15] J.L. Massey, D.J. Costello, J. Justesen, "Polynomial weights and code constructions", *IEEE Trans. Inform. Theory*, **IT-19**, pp. 101-110, Jan. 1973.

[16] J.L. Massey and S. Serconek, "Linear complexity of periodic sequences: a general theory", Advances in Cryptology - CRYPTO'96, N. Koblitz (Ed.), Lecture Notes in Computer Science Vol. 1109, pp. 358-371, Springer-Verlag, Berlin, 1996.

[17] H. Niederreiter, "Some computable complexity measures for binary sequences", Proc. SETA98, pp. 67-78, Springer Verlag, 1999.

[18] K.G. Paterson, "Perfect Maps", *IEEE Trans. Inform. Theory*, **IT-40**, pp. 743-753, 1994.

[19] M.J.B. Robshaw, "On Evaluating the Linear Complexity of a Sequence of Least Period 2^n", Designs, Codes and Cryptography **4**(3), pp. 263-269, 1994.

[20] R.A. Rueppel, *Analysis and Design of Stream Ciphers,* Springer-Verlag, Berlin, 1986.

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