Modified Berlekamp-Massey algorithm for approximating the k-error linear complexity of binary sequences

Modified Berlekamp-Massey Algorithm for Approximating the k-error Linear Complexity of Binary Sequences

Alexandra Alecu and Ana Sălăgean

Department of Computer Science,
Loughborough University,
Loughborough, UK,
{A.Alecu, A.M.Salagean}@lboro.ac.uk,
http://www.lboro.ac.uk

Abstract. Some cryptographical applications use pseudorandom sequences and require that the sequences are secure in the sense that they cannot be recovered by only knowing a small amount of consecutive terms. Such sequences should therefore have a large linear complexity and also a large k-error linear complexity. Efficient algorithms for computing the k-error linear complexity of a sequence only exist for sequences of period equal to a power of the characteristic of the field. It is therefore useful to find a general and efficient algorithm to compute a good approximation of the k-error linear complexity. We show that the Berlekamp-Massey Algorithm, which computes the linear complexity of a sequence, can be adapted to approximate the k-error linear complexity profile for a general sequence over a finite field. While the complexity of this algorithm is still exponential, it is considerably more efficient than the exhaustive search.

Keywords: pseudorandom sequences, stream ciphers, linear complexity, k-error linear complexity

1 Introduction

The k-error linear complexity of a sequence is a generalisation of the notion of linear complexity. While the linear complexity of a sequence is defined as the length of the smallest linear recurrence relation which generates that sequence, the k-error linear complexity is the length of the smallest linear recurrence relation which generates a sequence which differs from the original sequence in at most k positions.

When designing a stream cipher, the keystream sequence has to have a large linear complexity. Using the Berlekamp-Massey Algorithm, a sequence can be efficiently recovered by knowing a number of consecutive terms equal to twice its linear complexity. Sequences with low linear complexity would therefore be vulnerable to known plaintext attacks. Similarly, sequences with low k-error linear complexity for small values of k could also be vulnerable if the corresponding linear recurrence relation was found.

An exact algorithm to compute the k-error linear complexity only exists for periodic sequences over a finite field $GF(p^m)$ and with period a power of p, p being

prime and $m \geq 1$ (see Stamp and Martin [10], Lauder and Paterson [5] for $p = 2$ and Kaida, Uehara and Imamura [4] for an arbitrary p). These algorithms are based on the algorithms of Games and Chan [3] and Ding, Xiao, Shan [2] for computing the linear complexity of such sequences, and they work only when a full period of the sequence is known, i.e. the whole sequence is known, which is not the case in cryptanalysis applications.

We propose adapting the Berlekamp-Massey Algorithm (Berlekamp [1], Massey [7]) which computes the linear complexity, in order to approximate the k-error linear complexity profile for a general sequence over a finite field. The main idea is to devise a heuristic algorithm which explores only some of all the possible error patterns. The choice of the positions of the errors is guided by the steps of the Berlekamp-Massey Algorithm in which the complexity is increased.

While the proposed algorithm has an exponential complexity, the base of the exponential function is smaller than for an exhaustive search; moreover, the base for the proposed algorithm is independent of the size of the field, while for an exhaustive search it increases with the size of the field.

In this paper we consider mainly binary sequences but the proposed algorithm can be extended to arbitrary finite fields.

2 Background

Definition 1. *Given an infinite sequence $s = s_0, s_1, \ldots$ (or a finite sequence $s = s_0, s_1, \ldots, s_{t-1}$) with elements in a field K, we say that s is a* linear recurrent sequence *if it satisfies a relation of the form $s_j + c_{L-1}s_{j-1} + \ldots + c_1 s_{j-L+1} + c_0 s_{j-L} = 0$ for all $j = L, L+1, \ldots$ (or for all $j = L, L+1, \ldots t-1$, respectively), where $c_0, c_1, \ldots, c_{L-1} \in K$ are constants. The associated* characteristic polynomial *is $C(X) = X^L + c_{L-1}X^{L-1} + \ldots + c_1 X + c_0$. If L is minimal for the given sequence, we call L the* linear complexity *of s, denoted $L(s)$.*

The notion of linear complexity has been generalised to k-error linear complexity by Stamp and Martin [10] (see also Ding, Xiao, Shan [2]). In the following $w_H(s)$ denotes the Hamming weight i.e. the number of non-zero terms of s.

Definition 2. *For a given infinite sequence $s = s_0, s_1, \ldots$ of period N, with elements in a field K and for a fixed integer k, $0 \leq k \leq w_H((s_0, \ldots, s_{N-1}))$, the* k-error linear complexity *of the sequence s is defined as*

$$L_k(s) = \min\{L(s+e) |\ e \text{ is a sequence of period } N \text{ over } K, w_H((e_0, e_1, \ldots, e_{N-1})) \leq k\}$$

For a given finite sequence $s = s_0, s_1, \ldots, s_{t-1}$ with elements in a field K and for a fixed integer k, $0 \leq k \leq w_H(s)$, the k-error linear complexity *of the sequence s is defined as $L_k(s) = \min\{L(s+e) | e \in K^t, w_H(e) \leq k\}$. The sequences e are called error sequences or error patterns. The k-error linear complexity profile of the sequence is defined as being the set of pairs $(k, L_k(s))$, for all k with $0 \leq k \leq w_H(s)$.*

Property 1. Given a (finite or infinite) sequence s with elements in a finite field $GF(q)$, we have $L_i(s) \geq L_j(s)$, for all $i < j$.

The Berlekamp-Massey Algorithm ([1],[7]) computes the characteristic polynomial and the linear complexity of a sequence over any field. It is iteratively processing each term of a finite sequence $s_0, s_1, \ldots, s_{t-1}$, adjusting the characteristic polynomial when necessary. At each step n the current minimal characteristic polynomial $C^{(n)}(X)$ generates the n sequence terms s_0, s_1, ..., s_{n-1} processed so far. In addition, the last characteristic polynomial $C^{(m)}(X)$ of degree strictly smaller than the degree of $C^{(n)}(X)$ is also stored. We denote $L^{(i)} = \deg(C^{(i)})$. The discrepancy $d^{(n)}$

$$d^{(n)} = s_n + \sum_{i=0}^{L^{(n)}-1} c_i^{(n)} s_{i+n-L^{(n)}} \tag{1}$$

is the difference between the term which is expected using the current characteristic polynomial and the actual term s_n which is currently processed. Three possible cases are identified:

1. If $d^{(n)} \neq 0$ then s_n cannot be generated using $C^{(n)}(X)$:
   a) If $2L^{(n)} > n$ then the new characteristic polynomial is computed as $C^{(n+1)}(X) \leftarrow C^{(n)}(X) - \frac{d^{(n)}}{d^{(m)}} \cdot X^{(m-L^{(m)})-(n-L^{(n)})} \cdot C^{(m)}(X)$ and it has the same degree as the previous one;
   b) If $2L^{(n)} \leq n$ then the new characteristic polynomial is computed as $C^{(n+1)}(X) \leftarrow X^{(n-L^{(n)})-(m-L^{(m)})} \cdot C^{(n)}(X) - \frac{d^{(n)}}{d^{(m)}} \cdot C^{(m)}(X)$ and it has a higher degree than the previous one, namely $L^{(n+1)} = n + 1 - L^{(n)}$; m is updated to n.
2. If $d^{(n)} = 0$ then s_n can be generated using $C^{(n)}(X)$, so the characteristic polynomial stays unchanged $C^{(n+1)}(X) = C^{(n)}(X)$.

We initialise $C^{(i)}(X) \leftarrow 1$ for $i = 0, \ldots, j$, $C^{(j+1)}(X) \leftarrow X^{j+1}$ and $m \leftarrow j$, where s_j is the first non-zero term of the sequence. At the end of the algorithm, $L^{(t)}$ is the linear complexity of the sequence and $C^{(t)}(X)$ is a minimal characteristic polynomial (which is unique if $2L^{(t)} \leq t$, otherwise it may not be unique).

3 The Modified Berlekamp-Massey Algorithm

Determining the k-error linear complexity of a finite binary sequence of length t by an exhaustive search approach would mean investigating all the $\sum_{i=0}^{k} \binom{t}{i}$ patterns of up to k errors and computing the linear complexity of each of the sequences obtained by adding these error patterns to the original sequence. Some computational savings can be made by taking advantage of the incremental nature of the Berlekamp-Massey Algorithm; for error patterns which coincide on the first say i positions, reuse the computations made on the first i terms of the sequence. We implemented this more efficient version of an exhaustive search for the binary case and used it as a reference (we denote this algorithm the Efficient Exhaustive Search Algorithm).

A heuristic approach would only explore some of all the possible error patterns. Our proposed heuristic will use the Berlekamp-Massey Algorithm to choose these patterns. Namely, during the algorithm, only the case when the discrepancy $d^{(n)} \neq 0$ and $2L^{(n)} \leq n$ (case (1b) in Section 2) yields an increase in the current complexity of the sequence. It seems therefore natural to concentrate on what would happen if

the current term of the sequence, which creates this increase in complexity, would be changed in such a way as to make the discrepancy zero, and therefore make an increase in complexity unnecessary. If we made these changes to the sequence early in the algorithm, we would soon run out of the k allowed errors, and we would not be able to explore the effect of errors on later terms of the sequence. Whenever case (1b) occurs in the algorithm we do therefore consider both possibilities: changing the current term of the sequence, or not changing it, and we continue exploring both branches. A tree of recursive calls is thus obtained.

Our approach is not guaranteed to give the exact result for the k-error linear complexity, as the error pattern that decreases the complexity the most may well not have the errors in those positions suggested by the Berlekamp-Massey Algorithm. Since we investigate only some of all the possible error patterns, our results will always be larger or equal to the optimum ones. We investigate experimentally in Section 4 how close the approximation is to the actual k-error complexity. Unfortunately we were unable to prove a theoretical bound on the approximation quality.

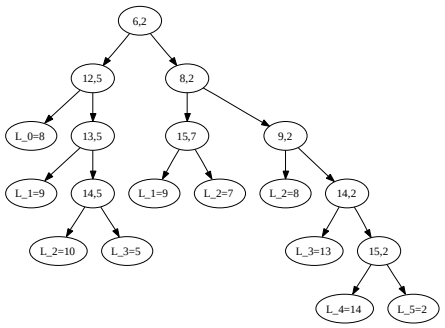

Fig. 1. Example of the Modified Berlekamp-Massey Algorithm tree of error and no-error recursive calls for the sequence $s = 0110111101110101$

We firstly illustrate our algorithm with an example:

Example 1. Applying the Berlekamp-Massey Algorithm to $s = 0110111101110101$ (length 16), the degree of the intermediate characteristic polynomials changes at positions 6 and 12 (ignoring the initial change from degree 0). The linear complexity is 8 and the characteristic polynomial $C(X) = X^8 + X^6 + X^5 + X^4 + X + 1$.

Figure 1 shows the tree of recursive calls for the MBM Algorithm. The internal nodes show the index of the element of the sequence currently processed and the current linear complexity. The left child of each internal node corresponds to not forcing an error and the right child corresponds to introducing the error. The leaves in the tree show the final result on each path in the tree: the number of errors which were introduced and the corresponding k-error linear complexity. In our example the first change of complexity happens when s_6 is processed. The error sequence can be built using a bottom-up technique. For example, for the 3-error linear complexity, the errors will be at indices 14,13 and 12, so the error sequence is 0000000000001110.

By taking the minimum value of the linear complexity for each number of errors, the results in the tree in Figure 1 give an incomplete approximate k-error linear complexity profile as being $\{(0,8),(1,9),(2,7),(3,5),(5,2)\}$. Applying Property 1 in Section 2 and using the fact that $L_{w_H(s)}(s) = 0$ the full approximate k-error linear complexity profile is found:
$\{(0,8),(1,8),(2,7),(3,5),(4,5),(5,2),(6,2),(7,2),(8,2),(9,2),(10,2),(11,0)\}$. The exact k-error linear complexity profile obtained by an exhaustive search algorithm is:
$\{(0,8),(1,7),(2,6),(3,4),(4,2),(5,1),(6,1),(7,1),(8,1),(9,1),(10,1),(11,0)\}$.

Algorithm 1 Recursive Modified Berlekamp-Massey Algorithm

Input: A finite non-zero sequence $s = s_0, s_1, \ldots, s_{t-1}$; k_{Max}; L_{Max}
Output: The approximate k-error linear complexity profile, sol
$v \leftarrow max(k_{Max}, w_H(s) - 1)$
for $i = 0, 1, \ldots, v$ **do**
    $sol_i \leftarrow \{t, X^t, (\underbrace{0, 0, \ldots, 0}_{t \text{ times}})\}$
end for
$k \leftarrow 0$
$e = (\underbrace{0, 0, \ldots, 0}_{t \text{ times}})$
$n \leftarrow 0$
while $s_n = 0$ and $n < t - 1$ **do** ▷ go over the initial zeros
    $n \leftarrow n + 1$
end while ▷ Initialize the details corresponding to 'last degree change' position m
$m \leftarrow n$
$C_m(X) \leftarrow 1$
$d_m \leftarrow s_n$
$n \leftarrow n + 1$
$C_n(X) \leftarrow X^n$ ▷ Initialize the details corresponding to current position n
call $mbmR(sol, m, C_m(X), d_m, C_n(X), n, k, e)$
return sol

We suppose that the input sequence has at least one non-zero element otherwise calculating the k-error linear complexity would be immediate since $L_k(\mathbf{0}) = 0$, for all $k \geq 0$, where $\mathbf{0}$ is an all-zero sequence of any length.

In practice we might be interested in the k-error complexities only for small values of k, say a certain percentage of the length of the sequence, so we introduce an extra parameter, k_{Max} (default value $w_H(s)$) in the algorithm and we compute the truncated k-error linear complexity profile $\{(i, L_i(s))|0 \leq i \leq k_{Max}\}$.

One might also be interested in the minimum number of errors needed to achieve a linear complexity below a certain set value L_{Max} (see Sălăgean [9]). This again would make some of the recursive calls unnecessary, when the current complexity is already equal to or below L_{Max}. The default value for L_{Max} is t. In this case the algorithm will return the profile $\{(i, \max\{L_i(s), L_{Max}\})|0 \leq i \leq k_{Max}\}$.

We implemented the algorithm in Algorithms 1 and 2 based on a recursive version of the Berlekamp-Massey Algorithm. Two new variables are needed to accommodate the current number of errors, k and the current error pattern, e. We denote the k-error linear complexity profile as sol and we define each element as a collection of three components $sol_k = \{sol\_L_k, sol\_C_k(X), sol\_err_k\}$, for $k = 0, 1, \ldots, \max\{k_{Max}, w_H(s) - 1\}$, where sol_L_k is the k-error linear complexity, $sol\_C_k(X)$ is the characteristic polynomial and sol_err_k is the error sequence corresponding to the k-error linear complexity.

There are some immediate improvements that can be performed on the Modified Berlekamp-Massey Algorithm. First of all the stop condition can be adjusted. Some of the paths taken by the recursion calls might get to a k-error linear complexity which is bigger or equal to the currently stored solution (sol_L_k) so they can be abandoned. Secondly, the currently stored k-error linear complexity profile can be maintained

Algorithm 2 The $mbmR$ procedure

procedure MBMR$(sol, m, C_m(X), d_m, C_n(X), n, k, e)$
  if $(n = t)$ or $(k > k_{Max})$ or $(\deg(C_n(X)) \leq L_{Max})$ **then**
    if $((n = t)$ and $(sol\_L_k > \deg(C_n(X)))$ and $k \leq k_{Max}$ **then**
      $sol_k \leftarrow \{\deg(C_n(X)), C_n(X), e\}$
    end if
  else
    $d_n \leftarrow (s + e)_n + \sum_{i=0}^{\deg(C_n(X))-1} c_n \cdot (s + e)_{i+n-\deg(C_n(X))}$
    if $d_n \neq 0$ **then**
      if $2 \cdot \deg(C_n(X)) > n$ **then** ▷ (1a) the complexity does not change
        $C_n(X) \leftarrow C_n(X) - \frac{d_n}{d_m} \cdot X^{(m-\deg(C_m(X)))-(n-\deg(C_n(X)))} \cdot C_m(X)$
        $mbmR(sol, m, C_m(X), d_m, C_n(X), n + 1, k, e)$
      else ▷ (1b) the complexity does change
        $T(X) \leftarrow C_n(X)$
        $C_n(X) \leftarrow X^{(n-\deg(C_n(X)))-(m-\deg(C_m(X)))} \cdot C_n(X) - \frac{d_n}{d_m} \cdot C_m(X)$
        $mbmR(sol, n, T(X), d_n, C_n(X), n + 1, k, e)$
        if $e_n = 0$ **then**
          $mbmR(sol, m, C_m(X), d_m, C_n(X)$,n+1$, k + 1, (e - d_n I_n))$
        end if
      end if
    else ▷ (2) the current characteristic polynomial does not change
      $mbmR(sol, m, C_m(X), d_m, C_n(X), n + 1, k, e)$
    end if
  end if
end procedure

whenever a new solution is found, using Property 1 in Section 2. Finally, we can combine iteration and recursion in order to minimize the stack size.

3.1 Algorithm Analysis

The correctness of the linear complexity and characteristic polynomial for each number of errors k and the corresponding error sequence stored in the solution array at the end of the algorithm, results from the way the algorithm was built and from the correctness of the Berlekamp-Massey Algorithm (Massey [7]).

For analysing the complexity of the algorithms we will use the trees described in Section 3 and estimate their number of nodes using the following Lemma:

Lemma 1. *A binary tree of depth n and with at most k right branches on any path from the root to a leaf has a maximum of $\sum_{i=0}^{k} \binom{n+1}{i+1}$ vertices.*

Proof. We associate to each node a description of the path from the root to that node, i.e. a sequence over the alphabet $\{L, R\}$, where L signifies a left branch and R a right branch. The number of nodes will therefore be equal to the number of sequences of lengths between 0 and n, each sequence containing at most k occurrences of R. For any fixed number i of R's we have

$$\sum_{m=i}^{n} \binom{m}{i} = \binom{n+1}{i+1}$$

such sequences, so the total for all i from 0 to k will be

$$\sum_{i=0}^{k} \binom{n+1}{i+1}.$$

There is no closed form for sums of the form $\sum_{i=0}^{k} \binom{n}{i}$, so we will use bounds:

Lemma 2. *The following bound stands*

$$\sum_{i=0}^{k} \binom{n}{i} \leq \begin{cases} 2\binom{n}{k}, & \text{if } k \leq \left\lfloor \frac{n+1}{3} \right\rfloor, \\ \left(k - \left\lfloor \frac{n+1}{3} \right\rfloor + 2\right) \binom{n}{k}, & \text{if } \left\lfloor \frac{n+1}{3} \right\rfloor < k \leq \left\lfloor \frac{n-1}{2} \right\rfloor \end{cases}$$

Proof. The first case follows by induction on k, using the fact that $\binom{n}{k} = \binom{n}{k-1} \cdot \frac{n-k+1}{k}$ for all n and k. Also $\frac{n-k+1}{k} \geq 2$ if $k \leq \lfloor (n+1)/3 \rfloor$. The remaining inequalities follow from the first using elementary properties of the binomial coefficients.

We will approximate binomial coefficients using the following (see [6, Lemma 7, Chapter 10])

$$\binom{n}{k} \approx c \frac{1}{\sqrt{n\alpha(1-\alpha)}} \left(\frac{1}{\alpha^{\alpha}(1-\alpha)^{1-\alpha}} \right)^{n} \tag{2}$$

where $0 < k < n$, $\alpha = k/n$ and c is a constant, $1/\sqrt{8} \leq c \leq 1/\sqrt{2\pi}$.

When assessing exponential time complexities of algorithms we will also use the fact that for any $a > 1$ and $i > 0$ we have $n^i a^n \in \mathcal{O}((a+\varepsilon)^n)$ with $\varepsilon > 0$ an arbitrarily small constant.

We are now ready to estimate the complexity of the algorithms presented.

Theorem 1. *The worst case time complexity of the Efficient Exhaustive Search Algorithm for sequences of length t and number of errors at most $k_{Max} = vt$ with $0 < v < 1/3$ is $\mathcal{O}(\sqrt{t}\lambda^t)$ where $\lambda = \frac{1}{v^v(1-v)^{1-v}}$. This can also be expressed as $\mathcal{O}((\lambda+\varepsilon)^t)$ with $\varepsilon > 0$ an arbitrarily small constant. For a typical value of $v = 0.1$ (i.e. errors in at most 10% of the positions) the time complexity is $\mathcal{O}(\sqrt{t}1.384145^t)$.*

Proof. The Efficient Exhaustive Search Algorithm will construct a tree of depth t and at most k_{Max} right branches on any path from the root to a leaf. By Lemma 1, this tree will have at most $\sum_{i=0}^{k_{Max}} \binom{t+1}{i+1}$ nodes. So the number of nodes is bounded by $2\binom{t+1}{k_{Max}+1}$, by Lemma 2. In any node we compute a discrepancy and possibly adjust the characteristic polynomial, so there are $\mathcal{O}(t)$ computational steps. Therefore the complexity is $\mathcal{O}(t\binom{t+1}{k_{Max}+1})$.

Using (2) we obtain the following approximation:

$$\begin{aligned} 2t\binom{t+1}{k_{Max}+1} &= 2\frac{t(t+1)}{k_{Max}+1}\binom{t}{k_{Max}} = \\ &= 2\frac{t(t+1)}{vt+1}\binom{t}{vt} \\ &\approx 2c\frac{t(t+1)}{vt+1}\frac{1}{\sqrt{tv(1-v)}}\left(\frac{1}{v^v(1-v)^{1-v}}\right)^t \\ &\approx \frac{2c}{v\sqrt{v(1-v)}}\sqrt{t}\left(\frac{1}{v^v(1-v)^{1-v}}\right)^t \end{aligned}$$

which is $\mathcal{O}(\sqrt{t}\lambda^t)$ where $\lambda = \frac{1}{v^v(1-v)^{1-v}}$.

For the Modified Berlekamp Massey Algorithm it is harder to estimate the depth of the tree, as the number of terms processed in between two decision points will vary depending on the particular sequence. We will assume that an average of u terms are processed between two decision points, i.e. between two points where the Berlekamp-Massey algorithm would prescribe an increase in the current complexity of the sequence. In [8, Chapter 4] it is shown that for random binary sequences the average number of bits that have to be processed between two changes in complexity is 4 and the change in complexity has an average of 2. While the sequences used in the cryptographic applications are not truly random, using a value of $u = 4$ for the average number of terms between two changes of complexity seems reasonable.

Theorem 2. *The worst case time complexity of the Modified Berlekamp Massey Algorithm for sequences of length t, an average of u terms of the sequence processed between two changes in complexity, and a number of errors at most $k_{Max} = vt$ with $0 < v < \frac{1}{u}$ is*

$$\begin{cases} \mathcal{O}(\sqrt{t}\lambda_1^t) & \text{if } v < \frac{1}{3u} \quad \text{where } \lambda_1 = \frac{1}{uv^v(1-uv)^{\frac{1}{u}-v}}, \\ \mathcal{O}(t\sqrt{t}\lambda_1^t) & \text{if } \frac{1}{3u} \le v < \frac{1}{2u} \quad \text{where } \lambda_1 = \frac{1}{uv^v(1-uv)^{\frac{1}{u}-v}}, \\ \mathcal{O}(t\lambda_2^t) & \text{if } \frac{1}{2u} \le v \le \frac{1}{u} \quad \text{where } \lambda_2 = \sqrt[u]{2}. \end{cases}$$

In all cases the complexity can also be written as $\mathcal{O}((\lambda_i + \varepsilon)^t)$ where $\varepsilon > 0$ is an arbitrarily small constant. For a typical value of $v = 0.1$ (i.e. errors in at most 10% of positions) and $u = 4$ the complexity is $\mathcal{O}(t\sqrt{t}1.189208^t)$.

Proof. Since u is the number of terms between two decision points and t is the total number of terms, the depth of the tree will be t/u. We bound the number of vertices in the tree by $\sum_{i=0}^{k_{Max}} \binom{\frac{t}{u}+1}{i+1}$, using Lemma 1. When the number of right branches on any path, k_{Max}, is at most half the depth of the tree, by applying the first or the second bound in Lemma 2 (depending on whether k_{Max} is smaller or greater than a third of t/u), followed by the estimation (2), we obtain the first two computational complexities $\mathcal{O}$ of the Theorem in a similar way as in the proof of Theorem 1.

When the number of right branches allowed in the tree approaches the depth of the tree, i.e. k_{Max} approaches t/u, we will bound the number of nodes by $2^{\frac{t}{u}+1} - 1$ (the number of nodes in a complete binary tree of depth t/u). Combining this with $\mathcal{O}(t)$ operations in each node gives the third $\mathcal{O}$ of the theorem.

The proposed algorithm has the advantage that even when the field has more than two elements, there are still only two choices that are investigated: introducing no error, or introducing an error of magnitude $-d^{(n)}$, where $d^{(n)}$ is the discrepancy; an exhaustive search approach would have to investigate all the possible error magnitudes for each error position, i.e. $\sum_{i=0}^{k} \binom{t}{i}(w-1)^i$ possibilities for a field of w elements. Both the complexities in Theorems 1 and 2 will increase by a factor of $(\log w)^2$ to account for the more costly operations in a field of w elements. However, the exponential part in the $\mathcal{O}$ estimate will remain unchanged in Theorem 2 (Modified Berlakamp-Massey Algorithm), whereas in Theorem 1 (Efficient Exhaustive Search), λ^t will be replaced by $(\lambda(w-1)^v)^t$.

For a typical value of $v = 0.1$ (i.e. errors in at most 10% of the positions) and an alphabet of $w = 16$ elements the worst case time complexity is $\mathcal{O}(\sqrt{t}1.826^t)$ for exhaustive search as compared to $\mathcal{O}(t\sqrt{t}1.189208^t)$ for the proposed modified Berlekamp-Massey algorithm.

4 Tests and Results

In order to estimate the efficiency and the accuracy of the algorithm, a comparison has been done between the optimised Modified Berlekamp-Massey (MBM) Algorithm and the Efficient Exhaustive Search (EES) Algorithm.

We define the accuracy, $ACC_k(s)$, as the ratio between $L_{MBM,k}(s)$, the approximate value of the k-error linear complexity obtained using the Modified Berlekamp-Massey Algorithm and $L_{EES,k}(s)$, the exact value obtained using the Efficient Exhaustive Search Algorithm, $L_{MBM,k}(s)/L_{EES,k}(s)$.

The running time improvement was computed as the ratio between the time taken by the Efficient Exhaustive Search Algorithm and the time taken by the Modified Berlekamp-Massey Algorithm on the same processor, $time_{EES}/time_{MBM}$.

The first test has involved running both algorithms on a number of 70 randomly chosen sequences of length 64 (each bit is generated with the C `rand()` linear congruential generator function).

Figure 2 presents the average, best and worst value of ACC_k over the 70 sequences tested. These results are detailed in Table 1 for $1 \leq k \leq 9$. For small values of k we notice that on average the k-error linear complexity obtained by the Modified Berlekamp-Massey Algorithm is pretty close to the actual value, being higher by only 3.37% for 1 error, increasing to 16.45% for 6 errors (i.e. errors in about 10% of the terms) and by 25.92% for 9 errors (i.e. about 15% of the terms). As k increases, the quality of the results obtained by the Modified Berlekamp-Massey Algorithm deteriorates. Note however that the small values of k are the ones of practical interest.

Table 1. The average accuracy of the results of the MBM Algorithm.

Number of errors k	1	2	3	4	5	6	7	8	9
Average ACC_k	1.03	1.06	1.09	1.11	1.14	1.16	1.19	1.22	1.25
Best ACC_k	1	1	1	1	1	1	1	1	1
Worst ACC_k	1.14	1.2	1.21	1.31	1.5	1.35	1.37	1.5	1.66

The average running time improvement was 12691, i.e. the MBM Algorithm was nearly 13000 times faster than the EES Algorithm. Even better time improvements are obtained when imposing limits for the number of errors and/or the maximum linear complexity. For example for k_{Max} equal to 15% of the length of the sequence and L_{Max} approx. 1/3 of the length, the time improvement is 24017.

A second experiment involved running the Modified Berlekamp-Massey Algorithm for sequences of different lengths. We used 20 random sequences for each even length between 8 and 64. The time improvement shows an exponential increase with the

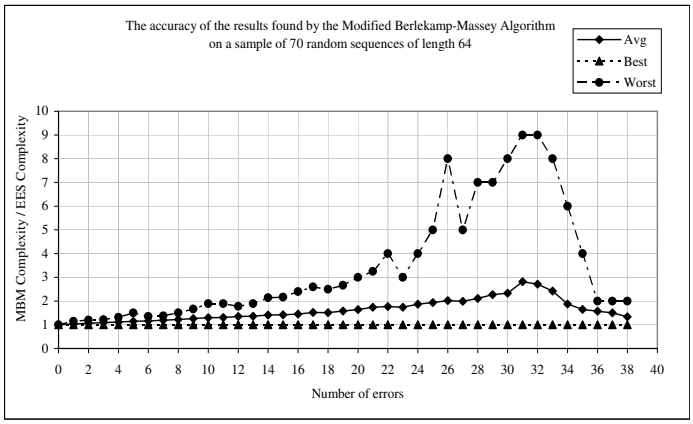

Fig. 2. The accuracy of the MBM Algorithm on a sample of 70 random sequences of length 64.

length of the sequence (no limitations were imposed on the parameters k_{Max} and L_{Max}). See figure 3 for the results.

The quality of the approximation was measured for each sequence at different levels of error: 5%, 10% and 15% of the length of the sequence. The results are summarised in figure 4. We note that the approximate value of the k-error complexity found by the modified Berlekamp-Massey Algorithm is consistently good on all lengths tested and it deteriorates only slightly as k increases as a percentage of the length of the sequence. For 5% errors (i.e. k is 5% of the length), the k-error linear complexity found by the MBM algorithm is on average not more than 10% higher than the actual value, for 10% errors it is at most 20% higher and for 15% it is at most 30% higher.

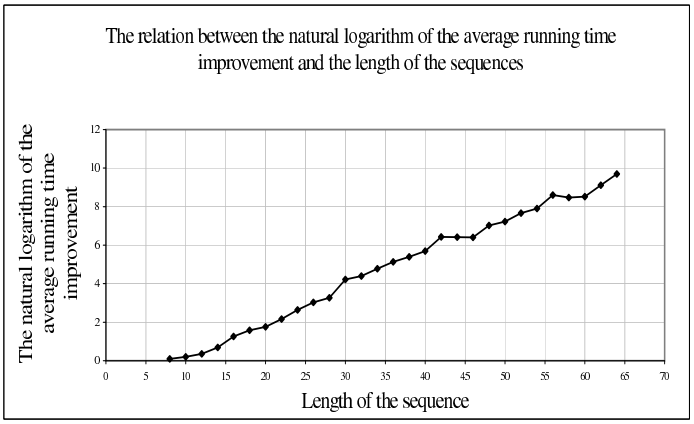

Fig. 3. The relation between the natural logarithm of the average running time improvement and the length of the sequences.

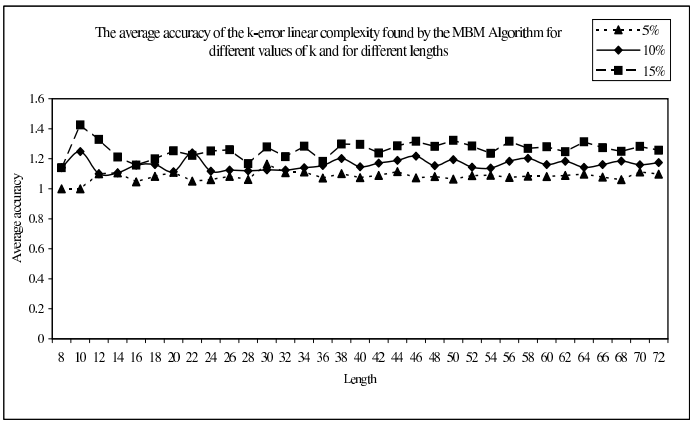

Fig. 4. The average accuracy of the k-error linear complexity found by the MBM Algorithm for different values of k and for different lengths.

For evaluating the accuracy of the MBM algorithm for sequences of higher length the actual k-error linear complexity could no longer be computed using exhaustive search due to hardware limitations. Instead, we carried out a controlled experiment where we took 50 sequences s of length 100, generated by a randomly chosen recurrence of size 33 (1/3 of the length). We computed the linear complexity $L(s)$ of each sequence s (this can be lower than 33). We artificially applied an error sequence e of weight k, such that the linear complexity of $s' = s + e$ is higher than $L(s)$. Obviously, $L_k(s') \leq L(s)$, so even though we do not know the exact k-error complexity of s', we do have a good upper bound. We then applied the MBM Algorithm to s' and computed the ratio $L_{MBM,k}(s')/L(s)$. This time the ratio can be less than 1 because $L(s)$ is an upper bound rather than the exact value of $L_k(s')$. Figure 5 presents the distribution of the values of this ratio in each interval of length 0.1. Four cases were considered, depending on the choice of k: random values up to 15% of the length of the sequence, or fixed values of 5%, 10% and 15%, respectively. We notice that a high proportion of the ratios are below 1.1, i.e. the value found by the MBM algorithm is close, or even lower than the original complexity, $L(s)$. The results improve when k represents a higher proportion of the length of the sequence.

5 Conclusion

We propose a heuristic algorithm for approximating the k-error linear complexity, based on the Berlekamp-Massey Algorithm. We implemented and tested this algorithm and the results are encouraging. The k-error linear complexity is approximated pretty close: on average it is only 16% higher than the exact value, for up to 6 errors on our test set of 70 random sequences of length 64. While the time complexity of the proposed algorithm is still exponential, it is considerably faster than an exhaustive search (on average about 13000 times faster for the sequences above). Even higher efficiency gains are expected in the non-binary case. Future work will investigate the possibility of further reducing the search space with minimal accuracy loss.

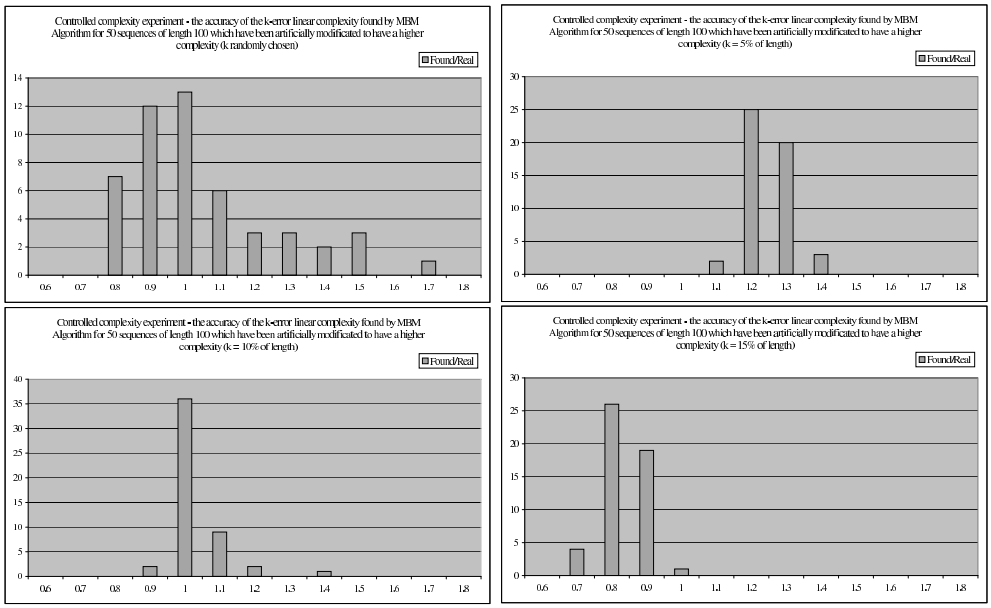

Fig. 5. The accuracy of the results found by MBM Algorithm on 50 sequences of length 100, when the sequences were artificially modified with errors sequences of weight : (a) random; (b) $k = 5\%$ of the length; (c) $k = 10\%$ of the length; (d) $k = 15\%$ of the length;

References

1. E. R. Berlekamp. *Algebraic Coding Theory*. McGraw-Hill, New York, 1968.
2. C. Ding, G. Xiao, and W. Shan. *The Stability Theory of Stream Ciphers*. Springer-Verlag, Heidelberg, 1992.
3. R. A. Games and A. H. Chan. A Fast Algorithm for Determining the Complexity of a Binary Sequence with Period 2^n. *IEEE Trans. Information Theory*, 29(1):144–146, 1983.
4. 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*, volume 151 of *Information and Computation*, pages 134–147. Academic Press, 1999.
5. A. G. B. Lauder and K. G. Paterson. Computing the Error Linear Complexity Spectrum of a Binary Sequence of Period 2^n. *IEEE Trans. Information Theory*, 49(1):273–2803, 2003.
6. F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-correcting Codes*. North Holland, Amsterdam, 1977.
7. J. L. Massey. Shift-Register Synthesis and BCH Decoding. *IEEE Trans. Information Theory*, 15(1):122–127, 1969.
8. R. A. Rueppel. *Analysis and Design of Stream Ciphers*. Springer-Verlag, New York, 1986.
9. A. Salagean. On the computation of the linear complexity and the k-error linear complexity of binary sequences with period a power of two. *IEEE Trans. Information Theory*, 51(3):1145–1150, 2005.
10. M. Stamp and C. F. Martin. An Algorithm for the k-Error Linear Complexity of Binary Sequences with Period 2^n. *IEEE Trans. Information Theory*, 39(4):1398–1401, 1993.