Computing the linear complexity for sequences with characteristic polynomial f^v

Noname manuscript No.
(will be inserted by the editor)

Computing the Linear Complexity for Sequences with Characteristic Polynomial f^v

Ana Sălăgean · Alex J. Burrage · Raphael C.-W. Phan

the date of receipt and acceptance should be inserted later

Abstract We present several generalisations of the Games-Chan algorithm. For a fixed monic irreducible polynomial f we consider the sequences s that have as a characteristic polynomial a power of f. We propose an algorithm for computing the linear complexity of s given a full (not necessarily minimal) period of s. We give versions of the algorithm for fields of characteristic 2 and for arbitrary finite characteristic p, the latter generalising an algorithm of Ding *et al.* We also propose an algorithm which computes the linear complexity given only a finite portion of s (of length greater than or equal to the linear complexity), generalising an algorithm of Meidl. All our algorithms have linear computational complexity. The proposed algorithms can be further generalised to sequences for which it is known a priori that the irreducible factors of the minimal polynomial belong to a given small set of polynomials.

Keywords Linear Complexity · Games-Chan Algorithm · Linear recurrent sequences

1 Introduction

The linear complexity of a sequence and the minimum linear recurrence relation (equivalently minimum linear feedback shift register(LFSR)) are important parameters for many applications, including cryptography.

Synchronous stream ciphers typically encrypt by xor-ing the plaintext with a pseudorandom keystream sequence generated from the key and initialisation vector. Known plaintext attacks can recover portions of this keystream sequence, and if the recovered portion consists of a number of successive terms equal to

A preliminary version of this paper was presented at the IEEE International Symposium on Information Theory, St. Petersburg, Russia, August 2011 [1]

Department of Computer Science, Loughborough University, UK, E-mail: A.M.Salagean@lboro.ac.uk · Qualcomm UK, E-mail: aburrage@qti.qualcomm.com · Faculty of Engineering, Multimedia University, Malaysia, E-mail: raphael@mmu.edu.my. This work was done while Alex J. Burrage and Raphael C.-W. Phan were with Loughborough University.

twice the linear complexity of the sequence, the whole keystream can be recovered efficiently (e.g. using the Berlekamp-Massey algorithm), effectively breaking the cipher. It is therefore essential to ensure that the keystream is a sequence with high linear complexity.

The well-known Berlekamp-Massey algorithm computes the linear complexity of a sequence in quadratic time. For certain classes of sequences more efficient algorithms exist. The Games-Chan algorithm [3] takes linear time and works for binary sequences with period of the form 2^n. It exploits the fact that in this case the minimal polynomial is a factor of $x^{2^n} - 1 = (x-1)^{2^n}$ hence it is a power of $x - 1$ and one only needs to determine which power. In Algorithm 1 we generalise the Games-Chan algorithm to the case when it is known a priori that the minimal polynomial is a power of a certain fixed irreducible polynomial f (so the Games-Chan algorithm would be the case $f = x - 1$).

The Games-Chan algorithm has been generalised to fields of arbitrary characteristic by Ding *et al.* in [2] and we will similarly give Algorithm 2 which is the generalisation of Algorithm 1 to arbitrary characteristic. (Our algorithm reduces to the one of Ding *et al.* in [2] when $f = x - 1$.)

The algorithms of Games-Chan and and Ding *et al.*, as well as our first two generalisations mentioned above assume that a whole (not necessarily minimal) period of the sequence is known. By contrast, Berlekamp-Massey algorithm only requires knowledge of a finite portion of consecutive terms of the sequence, of length equal to twice the linear complexity. The latter situation fits in with the cryptanalysis scenario mentioned earlier, where an attacker obtains a finite portion of the sequence and tries to determine the whole sequence. We will give therefore further generalisations of the algorithms of Games-Chan and and Ding *et al.* which will not require knowledge of the whole period.

It was noted by Sălăgean in [7] and by Meidl in [6] that it is actually not necessary to have a whole period of the sequence in order to determine its linear complexity using the Games-Chan algorithm. It suffices to have a number of terms greater or equal to the linear complexity, provided we still know that the sequence admits as a characteristic polynomial a power of $x - 1$ or more generally of some fixed irreducible polynomial f. For finite sequences which have a characteristic polynomial of the form f^v Meidl gives two algorithms in [6]: one for $f = x - 1$ and arbitrary v, the other for arbitrary f and v being a power of 2. We generalise his approach as Algorithm 4, which works for arbitrary f and arbitrary v. At first sight it would seem tempting to take this generalisation further, to k-error linear complexity, as in [6, Section 4]. However, we do not feel that such work would be worthwhile, as the definition used for the k-error complexity in [6] is a restricted one, (it computes the minimum linear complexity of all sequences z at Hamming distance k from s with the additional condition that z admits as a characteristic polynomial a power of f) and is not equivalent to the generally used definition (except for $f = x - 1$). A further explanation is contained in Remark 4.

All the algorithms mentioned so far have linear time complexity, like the Games-Chan algorithm, if we assume the irreducible polynomial is fixed.

We further generalise our algorithms to determine the minimal polynomial when all its irreducible factors are known a priori (Algorithm 5). This algorithm is efficient only if the number of irreducible factors is small and/or their weight is low.

For all the algorithms and their proofs we found it convenient to use the action of a polynomial on a sequence (Definition 3), a notion that has been used in different guises in several papers. We felt the proofs were shorter and simpler this way compared to the original proofs of many of the algorithms we generalised.

All the Games-Chan type algorithms, including those discussed here, derive their high efficiency from the fact that the minimal polynomial is a (possibly high) power of an irreducible polynomial (or of a small number of irreducible polynomials). We examine now where such sequences could arise in practice. Stream ciphers often use LFSR (Linear Feedback Shift Registers) and NFSR (Non-linear Feedback Shift Registers) as building blocks, and the period of the resulting sequence is derived from the period of these components. Recall that a sequence produced by an LFSR with n registers has period $2^n - 1$ or a factor thereof, which means its minimal polynomial (being a factor of $x^{2^n-1} - 1$) will only have distinct irreducible factors. On the other hand, a sequence produced by an NFSR has period 2^n (in which case it is called a de Bruijn sequence) or less, with no obvious divisibility relations between the possible periods (see [4, Section 3.1]). So the minimal polynomial of an NFSR sequence can have multiple factors, in fact the de Bruijn sequences have minimal polynomial $(x-1)^r$ for some r. Hence, algorithms like the ones in this paper would be relevant to some of the sequences obtained from NFSRs (or a combination of LFSRs and NFSRs) rather than LFSRs. Several of the recent stream cipher proposals (e.g. Grain, Trivium) contain NFSRs.

Algorithms like the ones in this paper, which determine the minimal polynomial when a characteristic polynomial is known, could also be potentially used in cryptography in the following scenario: it may be known from the theoretical analysis that the keystream has a particular characteristic polynomial, but this might not always be minimal, depending on the initial terms, which in turn depend of depending on the key and initialisation vector (IV) used. We would then want to detect the situations where the key/IV produce a sequence with a significantly lower linear complexity than expected.

2 Preliminaries

The linear complexity of a sequence is defined as usual:

Definition 1 Given an infinite sequence $s = s_0, s_1, \ldots$ (or a finite sequence $s = s_0, s_1, \ldots, s_{m-1}$) with elements in a field K we say that s is a *linear recurrent sequence* if it satisfies a *homogeneous linear recurrence relation*, i.e. there are constants $c_0, c_1, \ldots, c_{L-1} \in K$ such that

$$s_j + c_{L-1}s_{j-1} + \ldots + c_1 s_{j-L+1} + c_0 s_{j-L} = 0$$

for all $j = L, L+1, \ldots$ (or for all $j = L, L+1, \ldots m-1$, respectively). We associate to it a *characteristic polynomial* $g(x) = x^L + c_{L-1}x^{L-1} + \ldots + c_1 x + c_0$. If L is minimal for the given sequence, we call L the *linear complexity* of s and we call $g(x)$ a *minimal polynomial*.

For infinite sequences the minimal polynomial is unique and any other characteristic polynomial is a multiple of the minimal polynomial.

Throughout this paper we work in a field K of finite characteristic p. We denote by $s = s_0, s_1, \ldots$ an infinite sequence over K and by s' a finite sequence consisting

of successive terms of s. The infinite sequence s will be assumed to be periodic with period N, i.e. $s_i = s_{i+N}$ for all $i \geq 0$. We do not assume N is the minimum period. The finite or infinite sequence consisting only of zeroes will be denoted by $\mathbf{0}$ (the length of the sequence will be clear from the context). A monic irreducible polynomial $f \in K[x]$, different from 1 and x is fixed throughout. For a polynomial g we denote by $\mathrm{wt}(g)$ the weight of g, i.e. the number of non-zero coefficients of g (by analogy with the Hamming weight of vectors).

For convenience we will introduce the following notation:

Definition 2 Let g be a monic polynomial in $K[x]$. We define $\mathcal{M}(g)$ to be the set of all infinite sequences over K with a characteristic polynomial equal to g. We also define $\mathcal{M}(g^\infty) = \cup_{i=0}^{\infty} \mathcal{M}(g^i)$.

The following definition is a commonly used notion:

Definition 3 Let $g = \sum_{i=0}^{n} a_i x^i$ be a polynomial.

For an infinite sequence s we define the action of g on s, denoted gs, to be the infinite sequence $t = t_0, t_1, \ldots$ defined by $t_i = \sum_{j=0}^{n} a_j s_{i+j}$.

For a finite sequence $s' = (s_0, s_1, \ldots, s_{m-1})$ with $m > n$ we define the action of g on s', denoted gs', to be the finite sequence $t' = (t_0, t_1, \ldots, t_{m-n-1})$ defined by $t_i = \sum_{j=0}^{n} a_j s_{i+j}$. (One could extend the definition to $m \leq n$ but this situation will not occur in this paper).

Using the terminology of actions, the following results concerning characteristic polynomials are immediate:

Lemma 1 *Let $g \in K[x]$ be monic and let s be an infinite sequence. Then:*
(i) g is a characteristic polynomial of s iff $gs = \mathbf{0}$. Moreover, g is the minimal polynomial of s iff g is a polynomial of minimal degree for which $gs = \mathbf{0}$.
(ii) Let $h \in K[x]$ be monic. If g is the minimal polynomial of s then $g/\gcd(g,h)$ is the minimal polynomial of hs.
(iii) If s is periodic and N is a period of s then N is also a period of gs.

Proof Parts (i) and (iii) are clear. For (ii) let $\gcd(g,h) = g_2$ and $g = g_1 g_2$. Write h as $g_2 h'$ and denote the minimal polynomial of hs by g_3. We will prove that $g_1 = g_3$. From (i) we know $gs = g_1 g_2 s = \mathbf{0}$ and $g_3 hs = g_3 g_2 h' s = \mathbf{0}$. Since the minimal polynomial of a sequence divides any other characteristic polynomial, $g_1 g_2 | g_3 g_2 h'$. Since h' and g_1 are coprime we have $g_1 | g_3$. On the other hand, $gs = \mathbf{0}$ implies $gh's = g_1 g_2 h' s = g_1 hs = \mathbf{0}$ and since g_3 is the minimal polynomial of hs, we have $g_3 | g_1$, which combined with $g_1 | g_3$ proved previously results in $g_3 = g_1$. □

Based on the well-known exponentiation rule (sometimes known as "Freshman's dream"): $(a+b)^p = a^p + b^p$ for all $a, b \in K$, we have:

Lemma 2 *Let g be a polynomial over a field K of finite characteristic p. Then* $\mathrm{wt}(g^{p^w}) = \mathrm{wt}(g)$ *for any $w \geq 0$.*

Recall that the order of a polynomial g (with $x \nmid g$), denoted $\mathrm{ord}(g)$ is the smallest integer m such that g is a factor of $x^m - 1$. The minimal period of a periodic sequence is the same as the order of its minimal polynomial. The order of an irreducible polynomial f is $p^{\deg(f)} - 1$ or a factor thereof; hence $\mathrm{ord}(f)$ is not divisible by p. The order of a power of an irreducible polynomial can be derived as follows:

Theorem 1 *([5, Theorem 3.8]) Let f be irreducible over $K[x]$ and let r be a positive integer. Then* $\mathrm{ord}(f^r) = p^t \,\mathrm{ord}(f)$ *where t is the smallest integer with $p^t \geq r$.*

3 Linear Complexity of Infinite Sequences which have a Characteristic Polynomial equal to a Power of an Irreducible Polynomial

We assume that we are given an infinite sequence s of (not necessarily minimal) period N and that we know that the sequence admits as a characteristic polynomial a power of a fixed monic irreducible polynomial f, i.e. $s \in \mathcal{M}(f^\infty)$. Our goal is to determine the minimal polynomial of s, which will obviously be of the form f^r for some integer r. A naive algorithm could compute $f^i s$ for increasing values of i (by repeatedly replacing s by fs) until the zero sequence is obtained. Such an algorithm would have complexity $\mathcal{O}(rN)$ if f is considered fixed, or $\mathcal{O}(\mathrm{wt}(f)rN)$ if f is an input parameter. The more efficient method described here finds an upper bound for r and then does a p-ary search for the value of r. (We will see that this is indeed more efficient, see Theorem 3.)

An upper bound on the value of r can be obtained by looking at the period N and using Theorem 1:

Lemma 3 *Let $s \in \mathcal{M}(f^\infty)$ and let N be a (not necessarily minimal) period of s. Write N as $N = p^w N'$ with $p \nmid N'$. Then $s \in \mathcal{M}(f^{p^w})$,* $\mathrm{ord}(f) | N'$ *and* $p^w \,\mathrm{ord}(f)$ *is also a period of s.*

Proof Let f^r be the minimal polynomial of s and let w' be the smallest integer with $p^{w'} \geq r$. The minimal period of s is $\mathrm{ord}(f^r)$, which by Theorem 1 equals $p^{w'} \,\mathrm{ord}(f)$. Any other period of s, for example $N = p^w N'$ is a multiple thereof. Since neither $\mathrm{ord}(f)$ nor N' are divisible by p, this means $w' \leq w$ and $\mathrm{ord}(f) | N'$. Hence $r \leq p^{w'} \leq p^w$ so $s \in \mathcal{M}(f^{p^w})$ and $p^w \,\mathrm{ord}(f)$ is a period of s. □

Once we have an upper bound for r we can find the exact value of r by a p-ary search. We actually find the largest exponent i for which $f^i s \neq \mathbf{0}$, i.e. $r - 1$. To obtain r, a correction of $+1$ is added at the end. One can view the p-ary search equivalently as determining the digits of the base p representation of $r - 1$. When testing whether $f^i s \neq \mathbf{0}$ for different values of i, the values of i which are powers of p are preferred for efficiency reasons, see Lemma 2.

Lemma 4 *Let $s \in \mathcal{M}(f^\infty)$, $s \neq \mathbf{0}$ and let $N = p^w N'$ with $p \nmid N'$ be a (not necessarily minimal) period of s (which by Lemma 3 means $s \in \mathcal{M}(f^{p^w})$). Let f^r be the minimal polynomial of s. If $w = 0$ then $r = 1$. For $w \geq 1$, let $r - 1 = r_{w-1}p^{w-1} + r_{w-2}p^{w-2} + \ldots + r_1 p + r_0$ with $r_i \in \{0, 1, \ldots, p-1\}$ be the representation of $r - 1$ in base p. Then we have: r_{w-1} is the largest integer $i \geq 0$ for which $f^{ip^{w-1}} s \neq \mathbf{0}$.*
Moreover, putting $t = f^{r_{w-1}p^{w-1}} s$ we have $t \in \mathcal{M}(f^{p^{w-1}})$, t has minimal polynomial $f^{r - r_{w-1}p^{w-1}}$ and period N/p.

Proof It is easy to see that $r_{w-1}p^{w-1} \leq r - 1 < (r_{w-1}+1)p^{w-1}$, hence $r_{w-1}p^{w-1} < r \leq (r_{w-1}+1)p^{w-1}$. So $f^{r_{w-1}p^{w-1}}$ is not a characteristic polynomial of s, whereas $f^{(r_{w-1}+1)p^{w-1}}$ is. By Lemma 1(i), this means $f^{r_{w-1}p^{w-1}} s \neq \mathbf{0}$ and $f^{(r_{w-1}+1)p^{w-1}} s = \mathbf{0}$.

The last equality also means that $f^{p^{w-1}}t = \mathbf{0}$, i.e. $t \in \mathcal{M}(f^{p^{w-1}})$. By Theorem 1, $p^{w-1}\,\mathrm{ord}(f)$ is a period of t. By Lemma 3, we know $\mathrm{ord}(f)|N'$. Hence $p^{w-1}N' = N/p$ is also a period of t. □

From Lemmas 3 and 4 we have:

Corollary 1 *With the notations of Lemma 4:* $r_{w-1} = 0$ *iff* $f^{p^{w-1}}s = \mathbf{0}$ *iff* $s \in \mathcal{M}(f^{p^{w-1}})$ *iff* s *has period* N/p.

Based on the Lemmas and Corollary above we present the following two algorithms LinCompChar2 and LinComp given as Algorithms 1 and 2. The first is for $p = 2$ and the second for arbitrary p (including $p = 2$). We formulated the algorithm for $p = 2$ separately due to the importance of characteristic 2 and similarity with Games-Chan algorithm, as well as the fact that the code is somewhat simplified by the existence of only 2 cases at each step, corresponding to a 0/1 digit in the binary representation of $r - 1$, so the test in Corollary 1 is sufficient for determining the digit.

Note that throughout the algorithms the current value of the infinite sequence s is implicitly stored as being the finite sequence $s' = (s_0, s_2, \ldots, s_{N-1})$ of length N, repeated periodically. When computing the action of a polynomial say $g = \sum_{i=0}^{n} a_i x^i$ on the infinite s thus stored, the result will be an infinite sequence t of period N stored as the finite sequence $t' = (t_0, t_2, \ldots, t_{N-1})$ consisting of the first N terms of t, computed from s' as $t_i = \sum_{j=0}^{n} a_j s_{(i+j) \bmod N}$.

Algorithm 1: LinCompChar2(s', N, f)

Input : $f \in K[x]$ an irreducible polynomial over a field K of characteristic 2;
$s' = (s_0, \ldots, s_{N-1})$ a finite sequence over K consisting of the first N terms of an infinite sequence s of (not necessarily minimal) period N such that $s \in \mathcal{M}(f^\infty)$.
Output: The minimal polynomial of the sequence s

```
1  begin
2      C = 0;
3      if s' = 0 then
4          return(f^C);
5      end
6      w = the largest integer for which 2^w divides N;
7      Optionally, if ord(f) precomputed, set N = 2^w ord(f) and s' = (s_0, s_1, ..., s_{N-1});
8      while w ≥ 1 do
9          t' = f^{2^{w-1}} s' (as action on an infinite sequence);
10         if t' ≠ 0 then
11             s' = t';
12             C = C + 2^{w-1};
13         end
14         s' = (s_0, s_1, ..., s_{N/2-1});
15         w = w - 1;
16         N = N/2;
17     end
18     C = C + 1;
19     return(f^C);
20 end
```

Algorithm 2: LinComp(s', N, f)

Input : $f \in K[x]$ an irreducible polynomial over a field K of characteristic p;
$s' = (s_0, \ldots, s_{N-1})$ a finite sequence over K consisting of the first N terms of an infinite sequence s of (not necessarily minimal) period N such that $s \in \mathcal{M}(f^\infty)$.
Output: The minimal polynomial of the sequence s

```
1  begin
2      C = 0;
3      if s' = 0 then
4          return(f^C);
5      end
6      w = the largest integer for which p^w divides N;
7      Optionally, if ord(f) precomputed, set N = p^w ord(f)  and s' = (s_0, s_1, ..., s_{N-1});
8      while w ≥ 1 do
9          t' = f^{p^{w-1}} s' (as action on an infinite sequence);
10         while t' ≠ 0 do
11             s' = t';
12             C = C + p^{w-1};
13             t' = f^{p^{w-1}} s' (as action on an infinite sequence);
14         end
15         s' = (s_0, s_1, ..., s_{N/p-1});
16         w = w - 1;
17         N = N/p;
18     end
19     C = C + 1;
20     return(f^C);
21 end
```

Theorem 2 *Algorithms LinCompChar2 and LinComp (Algorithms 1 and 2) are correct and terminate.*

Proof Let r be such that f^r is the minimal polynomial of the input sequence, i.e. the expected output of the algorithm. For the correctness of both algorithms we can prove the following invariants. At the start and at the end of each run of the outer while loop, s' is a finite sequence of length N, and $s' \neq \mathbf{0}$. If we denote by s the infinite sequence of period N obtained by repeating s', we have $s \in \mathcal{M}(f^{p^w})$ and the minimal polynomial of s is f^{r-C}.

We outline the proofs for these invariants. The fact that s' is a finite sequence of length N is immediate. The fact that $s' \neq \mathbf{0}$ is obviously true before the while loop, due to the first if statement, line 3. If s' is non-zero at the start of the outer while loop it will stay non-zero throughout, as each new value of s' is always set to either the first N/p elements of s' (and by Corollary 1, s' consists in this case of p repeating identical sequences, which are therefore non-zero) or to a non-zero value of t'.

The fact that $s \in \mathcal{M}(f^{p^w})$ and the minimal polynomial of s is f^{r-C} holds before the outer while loop due to Lemma 3 and is then maintained due to to Lemma 4.

Finally, upon exiting the outer while loop, we have that $w = 0$ so $s \in \mathcal{M}(f)$. Since $s' \neq \mathbf{0}$, this means f is the minimal polynomial of s. On the other hand we saw that the minimal polynomial of s is f^{r-C}, therefore $1 = r - C$ at this point, so $C + 1$ is correctly returned as r.

The termination follows from the fact that the value of w is decreased by one at each run of the while loop. Additionally, for characteristic p, the inner while loop will run at most $p-1$ times, as we know that at the beginning of each run of the outer while loop we have $s \in \mathcal{M}(f^{p^w})$ hence $f^{p^w} s = \mathbf{0}$. □

Theorem 3 *The complexity of Algorithms LinCompChar2 and LinComp (Algorithms 1 and 2) is $\mathcal{O}(N)$ if we consider f to be fixed, or $\mathcal{O}(\mathrm{wt}(f)N)$ if f is an input parameter.*

Proof It suffices to show the second result. Let N_0 be the initial value of N, w_0 be the initial value computed for w and let $N' = N_0/p^{w_0}$. The (outer) while loop will run w_0 times, as shown in the proof of Theorem 2.

In the binary case, the complexity of each individual loop is dominated by the calculation of $t' = f^{2^{w-1}} s'$, a finite sequence representing the first $2^w N'$ terms of an infinite sequence of period $2^w N'$. The number of summands for each term is fixed by Lemma 2 as $\mathrm{wt}(f)$. So the number of arithmetic operations is $2^w N'(\mathrm{wt}(f) - 1)$. For characteristic p each run of the outer while loop consists of at most p computations of $t' = f^{p^{w-1}} s'$, each taking $(\mathrm{wt}(f) - 1)p^w N'$ steps.

In total we have $\sum_{w=1}^{w_0} 2^w N'(\mathrm{wt}(f)-1) = 2(2^{w_0}-1)N'(\mathrm{wt}(f)-1) < 2N_0 \,\mathrm{wt}(f)$ for the binary case and $(\mathrm{wt}(f) - 1)N' \sum_{w=1}^{w_0} p^{w+1} = (\mathrm{wt}(f) - 1)p^2 N' \frac{p^{w_0}-1}{p-1} < \frac{p^2}{p-1} \mathrm{wt}(f) N_0$ for arbitrary p. □

Alternative algorithms can be obtained for LinCompChar2 and LinComp (Algorithms 1 and 2) by using the last equivalence of Corollary 1. Namely, we can check immediately at the start of the outer while loop whether the current value of s' consists of p repeating copies of the same sequence. If this is the case we do not compute t' but skip to the instructions for updating the values of s', w and N at the end of the loop. The algorithms thus modified would have the same worst-case complexity but will behave slightly better for the case when $r-1$ has many 0's in its representation in base p. Since each digit has a $1/p$ chance of being 0, the savings will be more significant in characteristic 2. We present an alternative to Algorithm 1 as Algorithm 3.

Algorithm 3: LinCompAlternative(s', N, f)

Comment: Same as algorithm LinCompChar2 except that lines 8-17 are replaced by
while $w \geq 1$ **do**
  if $(s_0, s_1, \ldots, s_{N/2-1}) \neq (s_{N/2}, s_{N/2+1}, \ldots, s_{N-1})$ **then**
    $s' = f^{2^{w-1}} s'$ (as action on an infinite sequence);
    $C = C + 2^{w-1}$;
  end
  $s' = (s_0, s_1, \ldots, s_{N/2-1})$;
  $w = w - 1$;
  $N = N/2$;
end

Remark 1 For $f = x - 1$, Algorithm 1 reduces to the Games-Chan algorithm, [3]. Firstly, $\mathrm{ord}(f) = 1$, so we can put $N = 2^w$ in line 7. Secondly, computing $t' = (x-1)^{2^{w-1}} s' = (x^{2^{w-1}} - 1)s'$ for a sequence of period 2^w means t' is the

component-wise subtraction of the two halves of s' (i.e. t' is $L(s) - R(s)$ if $L(s)$ and $R(s)$ denote the left and right half of s, as in the notation used in the Games-Chan algorithm). Therefore checking whether $t' = \mathbf{0}$ will in this case mean checking whether the two halves of s' are identical. Algorithm 3 will also become virtually the same as Algorithm 1 and the same as Games-Chan algorithm in this situation. Similarly, for $f = x - 1$ Algorithm 2 reduces to the algorithm of Ding et al. [2].

Note that in the Games-Chan algorithm the final instruction $C = C + 1$ is done conditionally, only if $s' \neq \mathbf{0}$. If one deals at the start of the algorithm with the case of an all-zero input sequence (as we do), it is no longer necessary to check at the end if $s' \neq \mathbf{0}$, as this will always be the case (see the proof of Theorem 2).

Example 1 Let $K = \mathrm{GF}(2)$ and $f = x^3 + x + 1$. The sequence $s \in \mathcal{M}(f^\infty)$ has period $N = 28$ and its first 28 terms are $s' = 0000000\ 0101100\ 0010111\ 0111011$. The running of Algorithm 1 is described in Table 1.

Table 1 Example run for Algorithm 1

s'	w	$t = \mathbf{0}$?	C
0000000 0101100 0010111 0111011	2	No	2
0010111 0010111	1	Yes	2
0010111	0		
$C = C + 1$			3
return(f^3)			

4 Linear Complexity of Finite Sequences which have a Characteristic Polynomial equal to a Power of an Irreducible Polynomial

It was noticed by Sălăgean in [7] and by Meidl in [6] that we actually do not need to have the whole period of the infinite sequence in the Games-Chan algorithm in order to compute the linear complexity. In this section we generalise the idea of Meidl, [6, Sections 2 and 3], by using the initial terms of a sequence, and knowledge of a characteristic polynomial to treat it as an infinite sequence.

For a fixed polynomial g, an individual infinite sequence s with characteristic polynomial g is uniquely defined (within the class of all sequences with characteristic polynomial g) by its initial $\deg(g)$ terms. Can we decide if s admits a characteristic polynomial of lower degree just by examining these initial $\deg(g)$ terms?

Lemma 5 *Let s be an infinite sequence with characteristic polynomial $g = g_1 g_2$ with g_1, g_2 monic. Then:*
s has characteristic polynomial g_1 iff the finite sequence $s' = (s_0, \ldots, s_{\deg(g)-1})$ has characteristic polynomial g_1.

Proof The direct implication is obvious. Conversely, assume s' has characteristic polynomial g_1 i.e. $g_1 s' = (0, \ldots, 0)$, a finite sequence of $\deg(g) - \deg(g_1) = \deg(g_2)$ terms. Note this sequence also coincides with the first $\deg(g_2)$ terms of the infinite sequence $g_1 s$. By Lemma 1(i), $gs = g_2 g_1 s = \mathbf{0}$, so $g_1 s$ has characteristic polynomial

g_2. But then $g_1 s = \mathbf{0}$ as its first $\deg(g_2)$ terms are all zero and its linear complexity is at most $\deg(g_2)$. □

Consequently, if we are given s' as being the first $v\deg(f)$ terms of a sequence $s \in \mathcal{M}(f^v)$ we can check whether s admits some characteristic polynomial of lower degree, i.e. $f^{v'}$ with $v' < v$ by checking whether $f^{v'}s' = \mathbf{0}$. Again, a p-ary search will make it more efficient.

The algorithm LinCompFinite is given as Algorithm 4 and is similar to the Algorithm 2 in the previous section. Note that throughout the algorithm, the length of the current value of s' is $v\deg(f)$ for the current value of v.

Algorithm 4: LinCompFinite(s', v, f)

Input : A finite sequence s' consisting of the first $v\deg(f)$ elements of an infinite sequence $s \in \mathcal{M}(f^v)$ where $f \in K[x]$ is a fixed irreducible polynomial over a field K of characteristic p.
Output: The minimal polynomial of the sequence s

```
1  begin
2      C = 0;
3      if s' = 0 then
4          return(f^C);
5      end
6      w = the smallest integer such that v ≤ p^w;
7      while w ≥ 1 do
8          t' = f^{p^{w-1}} s' (as action on a finite sequence);
9          if t' ≠ 0 then
10             s' = t';
11             C = C + p^{w-1};
12             v = v - p^{w-1};
13             w = the smallest integer such that v ≤ p^w;
14         else
15             v = p^{w-1};
16             w = w - 1;
17             s' = (s_0, s_1, ..., s_{v deg(f)-1});
18         end
19     end
20     C = C + 1;
21     return(f^C);
22 end
```

Theorem 4 *Algorithm 4 is correct and terminates.*

Proof Let $s^{(0)}$ be the original value of the infinite input sequence s and let r be such that f^r is its minimal polynomial. For the correctness, we will show that throughout the algorithm s' consists of the first $v\deg(f)$ terms of $f^C s^{(0)}$ and $p^w \geq v \geq r - C \geq 1$, with w minimal such that $p^w \geq v$. Therefore s' in conjunction with the characteristic polynomial f^v correctly defines the infinite sequence $f^C s^{(0)}$. All these statements are obviously true before the while loop begins. We will assume they are true at the start of a run of the loop and show they are true at the end of the loop.

If the $t' \neq \mathbf{0}$ branch of the **if** is taken, then $v_{new} = v_{old} - p^{w_{old}-1} \geq r - C_{old} - p^{w_{old}-1} = r - C_{new}$, where the *old* and *new* indices refer to the value at the

Table 2 Example run for Algorithm 4

s'	w	v	$t = \mathbf{0}$?	C
010100001011010110	3	6	Yes	0
010100001011	2	4	No	2
001111	1	2	No	3
101	0	1		
$C = C + 1$				4
return(f^4)				

beginning and at the end of the loop. In addition, $r - C_{new} \geq 1$: if we assume the contrary, i.e. $r \leq C_{new}$, then $f^{C_{new}}$ would be a characteristic polynomial for $s^{(0)}$, so $f^{C_{new}} s^{(0)} = \mathbf{0}$. But then s'_{new}, consisting of the first terms of $f^{C_{new}} s^{(0)}$, would equal $\mathbf{0}$. On the other hand $s'_{new} = t' \neq \mathbf{0}$, contradiction.

If the else branch of the **if** is taken, we know $f^{p^{w_{old}-1}} s'_{old} = \mathbf{0}$. By Lemma 5, this means $f^{C_{old}} s^{(0)}$ has characteristic polynomial $f^{p^{w_{old}-1}} = f^{v_{new}}$, so $v_{new} \deg(f)$ of its initial terms are sufficient to determine the sequence. Moreover, since $f^{r-C_{old}}$ is the minimal polynomial of $f^C_{old} s^{(0)}$ we have $p^{w_{old}-1} \geq r - C_{old}$. Hence $v_{new} = p^{w_{old}-1} \geq r - C_{old} = r - C_{new}$.

Finally, upon exiting the while loop we have $w = 0$ and since $1 = p^w \geq v \geq r - C \geq 1$, it follows that $r - C = 1$, so $C + 1$ is correctly returned as r.

To show termination, note that v decreases throughout the algorithm. □

Theorem 5 *Algorithm 4 run on a sequence of length m has complexity $\mathcal{O}(m)$ for a fixed f, or $\mathcal{O}(m \operatorname{wt}(f))$ if f is an input parameter.*

Proof We can see that each run of the while loop takes $v \deg(f)(\operatorname{wt}(f) - 1)$ steps for the current value of v, which is upper bounded by p^w. In each run of the loop, the value of w is decreased or stays the same, but can only stay the same for at most p successive runs of the while loop. So if w_0 is the initial value of w, we have at most $\sum_{w=1}^{w_0} p^{w+1} \deg(f)(\operatorname{wt}(f) - 1) = \frac{p^2}{p-1}(p^{w_0} - 1) \deg(f)(\operatorname{wt}(f) - 1) \leq \frac{p^3}{p-1} v \deg(f) \operatorname{wt}(f)$ steps. □

Example 2 Let $K = GF(2)$ and $f = x^3 + x + 1$. The finite sequence $s' = 010100001011010110$ consists of the first $6 \deg(f) = 18$ terms of a sequence $s \in \mathcal{M}(f^6)$. The running of Algorithm 4 is described in the Table 2.

Remark 2 Berlekamp-Massey algorithm has quadratic complexity. When we consider f fixed, Algorithm 4 has linear complexity, so it is clearly more efficient. Let us compare the two algorithms when f is an input parameter. Assume the minimal polynomial of s will turn out to be f^r. Assume both algorithms are run on a finite sequence s of length m, so we need that the length m satisfies $m \geq v \deg(f)$ for Algorithm 4 (where v is some upper bound on r known apriori) and $m \geq 2r \deg(f)$ for the Berlekamp-Massey algorithm. The computational complexity will be $\mathcal{O}(m(\operatorname{wt}(f) - 1))$ for Algorithm 4 (see proof of Theorem 5) and $\mathcal{O}(m \deg(f^r))$ for the Berlekamp-Massey algorithm. (A finer bound for the latter would be $\mathcal{O}(m(z - 1))$ where z is an upper bound on the weight of the minimal polynomial as well as the weights of all the intermediate minimal polynomials of the sequence. We have $z - 1 \leq \deg(f^r)$, and in the worst case equality is attained, for example in characteristic 2, for $f = x + 1$ and $r = 2^n - 1$ one can check that

$\mathrm{wt}((x+1)^{2^n-1})-1 = \deg((x+1)^{2^n-1})$.) Since $\mathrm{wt}(f)-1 \leq \deg(f) \leq \deg(f^r)$, Algorithm 4 is in general more efficient than Berlekamp-Massey. However, we must note that this comes at the cost of requiring the a priori knowledge of a characteristic polynomial of s.

Remark 3 For $f = x - 1$, $p = 2$ and arbitrary v, our Algorithm 4 reduces to Algorithm 1 of [6]. For $f = x^2 + x + 1$ and v being a power of 2, it reduces to Algorithm 2 in [6] (which, as remarked at the end on Section 3 of [6] could be generalised to arbitrary f and v a power of 2).

Let us examine the relation between the algorithms in this section (where a finite portion of the sequence is known) and the ones in the previous section (where a whole period of the sequence is known, i.e. the whole infinite sequence is known). We could easily transform one problem into the other, namely, if we have a finite sequence we can generate the whole period using the given characteristic polynomial f^v (note that this process can take a number of steps exponential in the length of the original input finite sequence, see discussion further on). Conversely given an infinite sequence of period N we could restrict to the initial $p^w \deg(f)$ terms (with w maximal such that $p^w | N$), as f^{p^w} is guaranteed by Lemma 3 to be a characteristic polynomial of f. We compare now the complexities of the two types of algorithm. Algorithms 1, 2 and 3 of the previous section are $\mathcal{O}(N \,\mathrm{wt}(f))$, and if $\mathrm{ord}(f)$ is known, it is $\mathcal{O}(p^w \,\mathrm{ord}(f)\,\mathrm{wt}(f))$ (as N can be replaced by $p^w \,\mathrm{ord}(f)$, see line 7 in the algorithms). On the other hand, Algorithm 4 of this section has complexity $\mathcal{O}(v \deg(f)\,\mathrm{wt}(f))$, so if we know no better bound than $v = p^w$ for the exponent of f (as is the case in the previous section), this becomes $\mathcal{O}(p^w \deg(f)\,\mathrm{wt}(f))$. Since $\deg(f) \leq \mathrm{ord}(f) \leq p^{\deg(f)} - 1$ with both lower and upper bounds attained for particular values of f, it means that the algorithms of the previous section are no better, and potentially exponentially slower than the ones in this section (to clarify, all are linear in the size of the input, but the size of the input can be exponentially higher if we use the full period rather than the initial $v \deg(f)$ terms). So the algorithms of the previous section should be avoided in favour of the one in this section. We did nevertheless present the former as they are direct generalisations of the Games-Chan algorithm.

Remark 4 Since the algorithm developed here is a generalization of the ones given by Meidl in [6], it may seem natural to go further and adapt these algorithms into ones capable of determining the k-error linear complexity, as a generalization of the work carried out in [6], Section 4. However, we do not believe that such work would be worthwhile for the following reason:

In [7] the following definition is given for the k-error complexity of a finite sequence, z, of length t with respect to a set A of infinite sequences:

$$c_k(z, A) = \min\{c(s) | s \in A, \mathrm{wt}((s_0, s_1, \ldots, s_{n-1}) - z) \leq k\}$$

where $c(t)$ is the complexity of the infinite sequence s. This definition is used in [7] with the set A being the set of all sequences with period a power of 2. This is because, if it is known that the initial finite sequence was part of an infinite sequence s whose period N was a power of 2, introducing errors to this infinite sequence will not affect this property (because the error sequence will have the same period as s, i.e. a power of two and hence adding them to the sequence will again result in a sequence with period a power of two), and so $c_k(z, A) = c_k(z, \mathcal{S}_N)$

where $\mathcal{S}_N$ is the set of all binary sequences o period N. However, this is not always the case, and specifically, it is not the case in the way the definition is used in [6] Section 4.

In [6] Section 4, the same general definition is used, although now the set A is defined to be $\mathcal{M}((x^2+x+1)^{2^v})$. However, $s \in \mathcal{M}((x^2+x+1)^{2^v})$ does not imply $(s+e) \in \mathcal{M}((x^2+x+1)^{2^v})$, for all sequences e of the same period N as s. Hence in this case $c_k(s, \mathcal{M}((x^2+x+1)^{2^v})) \neq c_k(s, \mathcal{S}_N)$. Therefore while the algorithm presented in [6] Section 4 does calculate $c_k(s, \mathcal{M}((x^2+x+1)^{2^v}))$, this is not equal to the k-error linear complexity (in the classical sense) of the input sequence s. Therefore we do not feel it is worthwhile adapting the algorithm presented in this paper to the k-error linear complexity problem, as it would suffer from the same restriction.

5 Linear Complexity of Infinite Sequences which have a Characteristic Polynomial equal to a Product of Known Irreducible Factors

With a simple adjustment to the algorithms in Section 4, we can greatly increase their scope, so that they can be applied to any sequence provided each of the irreducible factors of a characteristic polynomial are known.

As a consequence of Lemma 1(ii) we have:

Corollary 2 *Assume that the minimal polynomial of a sequence s is of the form $f_1^{r_1} f_2^{r_2} \dots f_k^{r_k}$, with f_i distinct irreducible polynomials. Let $v_i \geq r_i$ for $i = 2, \dots, k$. Then $f_1^{r_1}$ is a minimal polynomial of the sequence $f_2^{v_2} \dots f_k^{v_k} s$.*

Therefore, if we know each of the irreducible factors of a characteristic polynomial of a sequence and we have an upper bound on the powers of each irreducible polynomial, we can use Corollary 2 and Algorithm 4 to successively determine the powers of each irreducible polynomial in the minimal polynomial. The resulting algorithm LinCompSet is presented as Algorithm 5.

Algorithm 5: LinCompSet$(s', m, \{(f_1, v_1), \dots, (f_k, v_k)\},)$

Input : s' a finite sequence consisting of the first m terms of an infinite sequence $s \in \mathcal{M}(f_1^{v_1} \dots f_k^{v_k})$, where $\{f_1, \dots, f_k\}$ is a given set of distinct irreducible polynomials, v_i are positive integers and $m \geq \sum_{i=1}^{k} p^{\lceil \log_p v_i \rceil} \deg(f_i)$.

Output: The minimal polynomial of s

```
1  begin
2      g = 1;
3      for i = 1, 2, ..., k do
4          w_i = ⌈log_p v_i⌉
5      end
6      for i = 1, 2, ..., k do
7          t' = the first v_i deg(f_i) terms of ∏_{j≠i} f_j^{p^{w_j}} s' (as action on finite sequences);
8          g = g * LinCompFinite(t', v_i, f_i);
9      end
10     return(g);
11 end
```

Theorem 6 *For a sequence of length m, and a fixed set $\{f_1, \ldots, f_k\}$, Algorithm 5 has complexity $\mathcal{O}(m)$. For a general set of k elements $\{f_1, \ldots, f_k\}$, the algorithm will have complexity $\mathcal{O}((km \sum_{i=1}^{k} \mathrm{wt}(f_i)))$.*

Proof In each of the k runs of the outer for loop, the computation of t' takes $m((\sum_{j=1}^{k}(\mathrm{wt}(f_j)-1)) - (\mathrm{wt}(f_i)-1))$ operations if we do a straightforward implementation. One can obtain a more efficient implementation by reusing intermediate results from the computation of t' for different values of i, but none of the versions we considered achieved an improved $\mathcal{O}$ complexity class. LinCompFinite has complexity $\mathcal{O}(v_i \,\mathrm{wt}(f_i) \deg(f_i))$, more precisely at most $\frac{p^2}{p-1}(p^{w_i}-1)\deg(f_i)(\mathrm{wt}(f_i)-1)$ steps (see the proof of Theorem 3). Since $m \geq p^{w_i} \deg(f_i)$, this gives a total of at most $m\frac{p^2}{p-1}\sum_{j=1}^{k}(\mathrm{wt}(f_j)-1)$ i.e. $\mathcal{O}(m\sum_{i=1}^{k}\mathrm{wt}(f_i))$ for each of the k loops. □

Note that Algorithm 5 is therefore efficient only if $\{f_1, \ldots, f_k\}$ has a small cardinality k and the total weight of its elements is small.

In the algorithm we assumed that the irreducible polynomials $f_1, \ldots, f_k$ as well as the bounds $v_1, \ldots, v_k$ are known apriori. If this is not the case, but we do know that s has period N, we can write $N = p^w N'$ with $p \nmid N'$ and we know that the minimal polynomial of s is a factor of $(x^{N'}-1)^{p^w}$, i.e. all the irreducible factors of the minimal polynomial are factors of $x^{N'}-1$ and have multiplicity at most p^w. Therefore we can put $\{f_1, \ldots, f_k\} = \{f \in K[x] | f \text{ irreducible factor of } x^{N'}-1\}$ and $v_i = p^w$ for all i. Consequently we would need $m = N$. In this case, the algorithm is efficient when N' is a relatively small constant, or when $x^{N'}-1$ has only a few factors, and their weight is small. Here is such an example:

Example 3 Consider a binary sequence s' of length m and $\{f_1, f_2, f_3\} = \{x+1, x^3+x^2+1, x^3+x+1\}$. Assume we are also given values for v_i. (Alternatively, assume we know s' is part of a sequence of period $N = 2^w * 7$, so we can put $v_i = 2^w$ and $\{f_1, f_2, f_3\} = \{x+1, x^3+x^2+1, x^3+x+1\}$, as $x^7-1 = (x+1)(x^3+x^2+1)(x^3+x+1)$.) The number of operations in Algorithm 5 will be about $16m$ (see proof of Theorem 6), whereas Berlekamp-Massey's algorithm would need $\deg(g)m$ operations, where g is the minimal polynomial of s. So in this example, and for large enough degree of g, Algorithm 5 is more efficient.

In the worst case Algorithm 5 becomes less efficient than Berlekamp-Massey's algorithm. The number of operations in Algorithm 5 is at least $k^2 m$. We examine cases where k is large. For the particular case when $N' = 2^n - 1$ is a (Mersenne) prime and $m = N = p^w N'$ with w a small constant, we have $k = \frac{N'-1}{n} + 1$ (the factors of $x^{2^n-1}-1$ are in this case $x-1$ and the $\varphi(2^n-1)/n = (2^n-2)/n$ primitive polynomials of degree n as in [5, Theorem 3.5]). Since in this case $k \approx \frac{N}{\log N}$, the complexity of the algorithm becomes in the worst case $\Omega(N^3/\log^2 N)$ i.e. more than quadratic.

6 Conclusion

We proposed algorithms for computing the linear complexity and minimal polynomial for sequences which admit as a characteristic polynomial a power of a fixed irreducible polynomial f. They work for any field of finite characteristic and we

do not necessarily need the whole period of the sequence. For $f = x - 1$ our algorithms reduce to the algorithms of Games-Chan [3], Ding *et al.* [2] and Meidl [6]. We can also apply our algorithms to the case where the characteristic polynomial is a product of powers of a small number of known irreducible polynomials. All our algorithms have linear computational complexity (when assuming the irreducible polynomials are fixed).

References

1. A. J. Burrage, A. Sălăgean and R. C.-W. Phan, "Linear Complexity for Sequences with Characteristic Polynomial f^v", *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, pp.688-692, St. Petersburg, Russia, 2011.
2. C. Ding, G. Xiao and W. Shan, *The Stability Theory of Stream Ciphers.* Springer Verlag, 1991.
3. R. Games and A. Chan, "A fast algorithm for determining the complexity of a binary sequence with period 2^n", *IEEE Trans. Information Theory*, vol. 29, pp. 144–146, 1983.
4. S. W. Golomb. *Shift Register Sequences.* Aegean Park Press, 1982.
5. R. Lidl and H. Niederreiter, *Introduction to Finite Fields and Their Applications.* Cambridge university Press, 1994.
6. W. Meidl, "How to determine linear complexity and k-error linear complexity in some classes of linear recurring sequences", *Cryptography and Communications*, vol. 1, pp. 117–133, 2009.
7. A. Sălăgean, "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*, vol. 51, pp. 1145–1150, 2005.