确定周期序列 k 错线性复杂度的一个快速算法

魏仕民
(淮北煤炭师范学院计算机科学与技术系,安徽淮北 235000)

摘　要: 文中提出 $GF(q)$ 上计算周期为 $2p^n$ 的序列 k-错线性复杂度的一个快速算法(这里 p 和 q 是素数,并且 q 是一个模 p^2 的本原根). 新算法的计算复杂度为 $O(N)$ (这里 N 是序列的周期).

关键词: 流密码; 序列; 线性复杂度; k-错线性复杂度
中图分类号: TN918. 1　　**文献标识码**: A　　**文章编号**: 0372-2112 (2004) 05-0705-04

An Efficient Algorithm for Determining the k-Error Linear Complexity of Periodic Sequences

WEI Shi-min
(*Department of Computer Science & Technique, Huaibei Coal Normal College, Huaibei, Anhui* 235000, *China*)

Abstract: An efficient algorithm for computing the k-error linear complexity of a sequence with period $2p^n$ over $GF(q)$ is presented, where p and q are primes, and q is a primitive root of modulo p^2. It is a generalization of an algorithm for determining the linear complexity of a sequence with period $2p^n$ presented by Wei, Xiao and Chen. The computation complexity of the new algorithm is $O(N)$, where N is the period of the sequence.

Key words: stream cipher; sequence; linear complexity; k-error linear complexity

1　引言

密钥序列的线性复杂度是流密码强度的一个重要度量指标. 但有的序列的线性复杂度极不稳定,即当改变这些序列周期的一位或几位时,其线性复杂度发生很大的变化. 例如一个周期为 N 的二元序列 s 的第一周期为 $s^N = (0, \cdots, 0, 1)$,其线性复杂度 $c(s^N) = N$,只要把最后的一位的 1 变成 0,则该序列的线性复杂度降低为 0,这种序列的线性复杂度是极不稳定的,用来作为密钥序列是不安全的.

因此,序列的线性复杂度的稳定性与序列的不可预测性是密切相关的. 我国学者早就注意到这个问题,因而率先创立了流密码的稳定性理论[1~3],并引入球体复杂度、重量复杂度等流密码稳定性度量指标,国外学者也引入类似球体复杂度的 k-错线性复杂度的概念[4]. 设 s 是有限域 $GF(q)$ 上周期为 N 的序列,当改变 s 的周期中至多 $k(0 \leq k \leq N)$ 位后,得到的所有序列的线性复杂度中最小的线性复杂度,称为 s 的 k-错线性复杂度,记为 $c_k(s)$,即 $c_k(s) = \min_{w_H(t) \leq k} \{c(s+t)\}$,这里 t 是周期为 N 的序列, $w_H(t)$ 表示 t 的一个周期中不为零的元素个数, $c(s)$ 表示 s 的线性复杂度. 而 s 的半径为 k 的球体复杂度定义为 $SC_k(s) = \min_{1 \leq w_H(t) \leq k} \{c(s+t)\}$.

任何一个序列的 k-错线性复杂度都可以反复运用 B-M 算法计算出来,但为了计算周期为 N 的二元序列的 k-错线性复杂度,要把这个算法运用 $\sum_{j=1}^{k} \binom{N}{j}$ 次. 对于周期为 N 的二元序列,虽然我们已有了一些特殊周期序列线性复杂度的快速算法,但如果没有一个有效的算法来计算 k-错线性复杂度,也得将这些快速算法运用 $\sum_{j=1}^{k} \binom{N}{j}$ 次. 即使 N 和 k 不是太大,计算量也是很大的. 在 Games-Chan 算法[5]的基础上, Stamp 和 Martin[4]提出一个确定周期为 2^n 的二元序列的 k-错线性复杂度的一个快速算法. 文献[6]将这个算法推广为计算 $GF(p^m)$ 上周期为 p^n 序列的 k-错线性复杂度. 在文献[7,8]中我们分别提出计算周期为 p^n 的二元序列的 k-错线性复杂度和计算 $GF(q)$ 上周期为 p^n 的序列的 k-错线性复杂度的快速算法. 本文提出一个确定 $GF(q)$ 上周期为 $2p^n$ 的序列 k-错线性复杂度的一个快速算法,这里 p 和 q 是素数,并且 q 是一个模 p^2 的本原根. 该算法推广了由魏仕民、肖国镇和陈钟在文献[9]中提出的确定周期为 $2p^n$ 序列的线性复杂度的快速算法.

2　算法和证明

设 s 是 $GF(q)$ 上周期为 $N = 2p^n$ 序列, p 和 q 是奇素数,

收稿日期:2003-05-06;修回日期:2003-11-05
基金项目:国家自然科学基金(No. 60172015);安徽省自然科学基金(No. 03042204);安徽省教育厅自然科学研究计划项目(No. 2004kj317);国家重点基础研究规划发展项目(No. G1999035804)

并且 q 是一个模 p^2 的本原根，s^N 是 s 的第一个周期. 本节我们提出一个新的快速算法用来计算 s 的 k-错线性复杂度. 新算法基于下列算法.

算法 1[9] 初始值：$a \leftarrow s^N$，$l \leftarrow p^n$，$c \leftarrow 0$，$f \leftarrow 1$.

(1) 如果 $l = 1$，转向 (2)；否则 $l \leftarrow l/p$，$A_i = (a_{(i-1)l}, a_{(i-1)l+1}, \cdots, a_{il-1})$，$i = 1, 2, \cdots, 2p$，$B_j = (A_{2j-1}, A_{2j})$，$j = 1, 2, \cdots, p$，转向 (5).

(2) 如果 $a = (0, 0)$，停止；否则转向 (3).

(3) 如果 $a_0 = a_1$，则 $c \leftarrow c + 1$，$f \leftarrow (1 - x) f$，否则转向 (4).

(4) 如果 $a_0 + a_1 = 0$，则 $c \leftarrow c + 1$，$f \leftarrow (1 + x) f$，停止；否则 $c \leftarrow c + 2$，$f \leftarrow (1 - x^2) f$，停止.

(5) 如果 $B_1 = \cdots = B_p$，则 $a \leftarrow B_1$，转向 (1)；否则转向 (6).

(6) 如果 $A_{p+1} + A_1 = \cdots = A_{2p} + A_p$，则 $c \leftarrow c + (p - 1) l$，$f \leftarrow f\Phi_{2pl}(x)$，$a \leftarrow \left(\sum_{i=1}^{p} (-1)^{i+1} A_i, \sum_{i=1}^{p} (-1)^{i+1} A_{i+1} \right)$，转向 (1)；否则转向 (7).

(7) 如果 $A_{p+1} - A_1 = \cdots = A_{2p} - A_p$，则 $c \leftarrow c + (p - 1) l$，$f \leftarrow f\Phi_{pl}(x)$，$a \leftarrow (\sum_{i=1}^{p} A_i, \sum_{i=1}^{p} A_{i+1})$，转向 (1)；否则 $f \leftarrow f\Phi_{pl}(x)\Phi_{2pl}(x)$，$c \leftarrow c + 2(p - 1) l$，$a \leftarrow B_1 + \cdots + B_p$，转向 (1).

最终可得 s 的线性复杂度 $c(s) = c$，s 的极小多项式 $f_s(x) = f$.

利用算法 1，我们可以设计一个计算 s 的 k-错线性复杂度的快速算法如下.

算法 2 初始值：$a \leftarrow s^N$，$l \leftarrow p^n$，$c \leftarrow 0$，$\mathrm{cost}[a_i, a_i] \leftarrow 0$，当 $0 \le h \le q - 1$ 且 $h \ne a_i$ 时，$\mathrm{cost}[a_i, h] \leftarrow 1$，这里 $i = 0, 1, 2, \cdots, 2l - 1$，$K \leftarrow k$.

(1) 如果 $l = 1$，则 $T = \mathrm{cost}[a_0, 0] + \mathrm{cost}[a_1, 0]$，转向 (2)；否则，$l \leftarrow l/p$，$A_t = (a_{(t-1)l}, \cdots, a_{tl-1})$，$t = 1, \cdots, 2p$，$T[a_i, h] = \sum_{j=0}^{p-1} \mathrm{cost}[a_{i+2jl}, h]$，$h = 0, 1, \cdots, q - 1$，$T_{a_i} = \min_{0 \le h \le q-1} \{T[a_i, h]\}$，$i = 0, 1, \cdots, 2l - l$，$T = \sum_{i=0}^{2l-1} T_{a_i}$，转向 (4).

(2) 如果 $T \le K$，停止；否则，$T = \min_{0 \le h \le q-1} \{\mathrm{cost}[a_0, h] + \mathrm{cost}[a_1, h]\}$，转向 (3).

(3) 如果 $T \le K$，则 $c \leftarrow c + 1$，停止；否则，$c \leftarrow c + 2$，停止.

(4) 如果 $T \le K$，则 $K \leftarrow K - T$，$\mathrm{cost}[a_i, h] \leftarrow T[a_i, h] - T_{a_i}$，$i = 0, 1, \cdots, 2l - 1$，$h = 0, 1, \cdots, q - 1$，转向 (5)；否则，$B \leftarrow (A_{p+1} + A_1, \cdots, A_{2p} + A_p)$，$C \leftarrow (A_{p+1} - A_1, \cdots, A_{2p} - A_p)$，对 $h = 0, 1, \cdots, q - 1$，$i = 0, 1, \cdots, pl - 1$，$\mathrm{cost}[b_i, b_i + h] = \min_{d_1 + d_2 = h} \{\mathrm{cost}[a_i, a_i + d_1] + \mathrm{cost}[a_{i+pl}, a_{i+pl} + d_2]\}$，$\mathrm{cost}[c_i, c_i + h] = \min_{d_2 - d_1 = h} \{\mathrm{cost}[a_i, a_i + d_1] + \mathrm{cost}[a_{i+pl}, a_i + p_l + d_2]\}$；$T[b_i, h] = \sum_{j=0}^{p-1} \mathrm{cost}[b_{i+jl}, h]$，$T[c_i, h] = \sum_{j=0}^{p-1} \mathrm{cost}[c_{i+jl}, h]$，$h = 0, 1, \cdots, q - 1$，$T_{b_i} = \min_{0 \le h \le q-1} \{T[b_i, h]\}$，$T_{c_i} = \min_{0 \le h \le q-1} \{T[c_i, h]\}$，$i = 0, 1, \cdots, l - 1$，$T^B = \sum_{i=0}^{l-1} T_{b_i}$，$T^C = \sum_{i=0}^{l-1} T_{c_i}$，转向 (6).

(5) 做循环：对 $i = 0, 1, \cdots, 2l - 1$，当 $T[a_i, h] = T_{a_i}$ 时 $a_i = h$. 做完循环做 $a \leftarrow (A_1, A_2)$，转向 (1).

(6) 如果 $\min\{T^B, T^C\} \le K$，则 $K \leftarrow K - \min\{T^B, T^C\}$，$c \leftarrow c + (p - 1) l$，转向 (7)；否则，$c \leftarrow c + 2(p - 1) l$，$\mathrm{cost}[a_i, a_i + h] \leftarrow \min_{\sum_{j=0}^{p-1} d_j = h} \left\{ \sum_{j=0}^{p-1} \mathrm{cost}[a_{i+2jl}, a_{i+2jl} + d_j] \right\}$，$i = 0, 1, \cdots, 2l - 1$，$h = 0, 1, \cdots, q - 1$，$a \leftarrow \left(\sum_{i=1}^{p} A_{2i-1}, \sum_{i=1}^{p} A_{2i} \right)$，转向 (1).

(7) 如果 $T^B = \min\{T^B, T^C\}$，则 $a \leftarrow (\sum_{i=1}^{p} (-1)^{i+1} A_i, \sum_{i=1}^{p} (-1)^{i+1} A_{i+1})$，$\mathrm{cost}[a_i, a_i + h] \leftarrow \min_{\sum_{j=1}^{p} (-1)^{j+1} h_{i,j-1} = h} \{\sum_{j=1}^{p} \mathrm{cost}[a_{i+(j-1)l}, a_{i+(j-1)l} + h_{i,j-1}] \mid b_i = a_{i+(j-1)l} + h_{i,j-1} + a_{i+(j+p-1)l} + h_{i,j+p-1}, j = 1, 2, \cdots, p\} - \sum_{j=1}^{p} \mathrm{cost}[a_{i+(j-1)l}, d_{i+(j-1)l,b}]$，$i = 0, 1, \cdots, 2l - 1$，$h = 0, 1, \cdots, q - 1$，转向 (1). 否则，$a \leftarrow \left(\sum_{i=1}^{p} A_i, \sum_{i=1}^{p} A_{i+1} \right)$，$\mathrm{cost}[a_i, a_i + h] \leftarrow \min_{\sum_{j=1}^{p} h_{i,j-1} = h} \{\sum_{j=1}^{p} \mathrm{cost}[a_{i+(j-1)l}, a_{i+(j-1)l} + h_{i,j-1}] \mid c_i = a_{i+(j-1)l} + h_{i,j-1} - a_{i+(j+p-1)l} - h_{i,j+p-1}, j = 1, 2, \cdots, p\} - \sum_{j=1}^{p} \mathrm{cost}[a_{i+(j-1)l}, d_{i+(j-1)l,c}]$，$i = 0, 1, \cdots, 2l - 1$，$h = 0, 1, \cdots, q - 1$，转向 (1).

最终可得 s 的 k-错线性复杂度 $c_k(s) = c$. 这里 $\mathrm{cost}[a_i, h]$ 表示改变 a_i 为 h 所要付出的代价，即表示改变 a_i 为 h 所需要至少改变原始序列的位数.

我们首先根据域 $GF(q)$ 上运算次数来度量算法 2 的计算复杂度. 一次域运算是指域中两个元素进行一次加法、减法、乘法、除法或比较. 已经知道算法 1 的计算复杂度是 $O(N)$ ($N = 2p^n$)[9]. 我们只要度量算法 2 中计算改变元素的代价而付出的计算复杂度即可. 现在我们在证明算法 2 的同时，计算算法 2 中计量改变元素的代价而付出的计算复杂度.

定理 1 设 s 是 $GF(q)$ 上一个周期为 $N = 2p^n$ 的序列，q 是一个模 p^2 的本原根，且 $0 \le k \le p^n$. 则算法 2 在 n 步内计算出的 c 即是 s 的 k-错线性复杂度，并且算法 2 的计算复杂度为 $O(N)$ ($N = 2p^n$).

证明 显然，当 $k = 0$ 时，算法 2 退化为算法 1，我们在文献[9]中已给出了证明. 当 $k > 0$ 时，为了尽可能降低复杂度 c 我们允许在 s 中作不多于 k 位的改变. 与在算法 1 中一样，只有当 $(A_1, A_2) = (A_3, A_4) = \cdots = (A_{2p-1}, A_{2p})$ 不成立时 c 才增加. 因此，在算法 2 的第 m 步，当 $(A_1, A_2) = (A_3, A_4) = \cdots = (A_{2p-1}, A_{2p})$ 不成立时，如果能迫使 $(A_1, A_2) = (A_3, A_4) = \cdots = (A_{2p-1}, A_{2p})$，我们就这样做，因为这将避免 $2(p - 1) p^{n-m}$ 加到 c 中，而其余各步全部可能加到 c 中的是 $2p^{n-m}$.

假设现在我们计算到第 m 步，$\mathrm{cost}[a_i, h]$ 的值正确地给出了改变 a_i 为 h 所要付出的代价. 则 $l = p^{n-m}$，如果 $a_{i+2(j-1)l}$，$j = 1, 2, \cdots, p$，不全相等，则要将所有 $a_{i+2(j-1)l}$ 改变为一个相同的 h，迫使它们都相等，因而迫使它们都相等的代价就是改变所有 $a_{i+2(j-1)l}$ 为 h 的代价 $T[a_i, h] = \sum_{j=1}^{p-1} \mathrm{cost}$

$[a_{i+2jl}, h]$中的最小值 $T_{a_i} = \min_{0\le h\le q-1}\{T[a_i, h]\}$. 因此,算法 2 中变量 $\sum_{i=0}^{2l-1} T_{a_i}$ 正确给出了迫使 $(A_1, A_2) = (A_3, A_4) = \cdots = (A_{2p-1}, A_{2p})$ 的总代价. 由于计算所有的 $T[a_i, h]$需要 $2l(p-1)q$ 次加法;计算所有的 T_{a_i}需要 $2l(q-1)$ 次比较;计算 T 需要 $2l-1$ 次加法,所以计算 T 需要总的域运算次数为 $2l(p-1)q+2l(q-1)+2l-1=2lpq-1$.

当 $T\le K$ 时. ①如果 $a_{i+2(j-1)l}, j=1,2,\cdots,p$,不全相等,不妨 $T[a_i, d]=T_{a_i}$,则我们要将所有 $a_{i+2(j-1)l}$改变成 d,因为这样可以付出较低的代价迫使 $a_{i+2(j-1)l}$都相等. 注意到在这一步结束时将有 $a\leftarrow(A_1, A_2)$,如果我们要在第 $m+1$ 步改变 a_i 的值(在第 m 步 a_i 已被赋值为 d)为 h,为了保持在第 m 步 $a_{i+2(j-1)l}, j=1,2,\cdots,p$,都相等,我们必须在第 m 步将 $a_{i+2(j-1)l}$都变成 h,这将多付出代价 $T[a_i, h]=T_{a_i}$. 所以在这种情况下,我们正确计算了 $\mathrm{cost}[a_i, h]$. ②如果 $a_{i+2(j-1)l}, j=1,2,\cdots,p$,全相等,则 $T_{a_i}=0$. 注意到在这一步结束时将有 $a\leftarrow(A_1, A_2)$,如果我们要在第 $m+1$ 步改变 a_i 的值为 h,为了保持在第 m 步 $a_{i+2(j-1)l}$都相等,我们必须在第 m 步将 $a_{i+2(j-1)l}$都改变成 h,因而要多付出的代价为 $T[a_i, h]-T_{a_i}$. 因此,在这种情况下,我们正确计算了 $\mathrm{cost}[a_i, h], i=0,1,2,\cdots,2l-1$;并且计算所有 $\mathrm{cost}[a_i, h]$需要 $2lq$ 次减法.

当 $T>K$ 时,我们无法迫使 $(A_1, A_2)=(A_3, A_4)=\cdots=(A_{2p-1}, A_{2p})$,但有可能迫使 $A_{p+1}+A_1=\cdots=A_{2p}+A_p$ 或者 $A_{p+1}-A_1=\cdots=A_{2p}-A_p$,为方便,记 $B_i=A_i+A_{p+i}$, $C_i=A_i-A_{p+i}$, $i=1,2,\cdots,p$, $B=(B_1,\cdots,B_p)$, $C=(C_1,\cdots,C_p)$. 如果能迫使 $B_1=\cdots=B_p$ 或者 $C_1=\cdots=C_p$,我们就这样做,因为这将避免 $(p-1)p^{n-m}$ 加到 c 中,而其余各步全部可能加到 c 中的是 $2p^{n-m}$.

现在我们首先要计算改变 B 或 C 中的一位所要付出的代价. 我们如果要改变 b_i 的值为 b_i+h,我们只要分别改变 a_i 和 a_{i+pl}为 a_i+d_1 和 $a_{i+pl}+d_2$,这里 $d_1+d_2=h$. 所以,改变 b_i 为 b_i+h 的代价是改变 a_i 为 a_i+d_1 的代价与改变 a_{i+pl}为 $a_{i+pl}+d_2$ 的代价和中的最小值,即

$$\mathrm{cost}[b_i, b_i+h]=\min_{d_2-d_1=h}\{\mathrm{cost}[a_i, a_i+d_1]+\mathrm{cost}[a_{i+pl}, a_{i+pl}+d_2]\},\quad i=0,1,2,\cdots,pl-1.$$

我们正确计算了 $\mathrm{cost}[b_i, b_i+h]$. 这里计算 $\mathrm{cost}[b_i, b_i+h]$需要 q 次加法和 $q-1$ 次比较,因而计算所有的 $\mathrm{cost}[b_i, b_i+h]$需要 $(2q-1)pql$ 次域运算. 类似的讨论将说明我们正确地计算了改变 c_i 的值为 c_i+h 的代价为

$$\mathrm{cost}[c_i, c_i+h]=\min_{d_2-d_1=h}\{\mathrm{cost}[a_i, a_i+d_1]+\mathrm{cost}[a_{i+pl}, a_{i+pl}+d_2]\},\quad i=0,1,2,\cdots,pl-1.$$

并且计算所有的 $\mathrm{cost}[c_i, c_i+h]$需要 $(2q-1)pql$ 次域运算.

其次,我们考虑 B 和 C,如果 $b_{i+(j-1)l}, j=1,2,\cdots,p$,不全相等,则要将所有 $b_{i+(j-1)l}$改变为一个相同的 h,迫使它们都相等,因而迫使它们都相等的代价就是改变所有 $b_{i+(j-1)l}$ 为 h 的代价 $T[b_i, h]=\sum_{j=0}^{p-1}\mathrm{cost}[b_{i+jl}, h]$中的最小值 $T_{b_i}=\min_{0\le h\le q-1}\{T[b_i, h]\}$. 因此,算法 2 中变量 $T^B=\sum_{i=0}^{l-1}T_{b_i}$ 正确给出了迫使 $B_1=\cdots=B_p$ 的总代价. 这里计算所有 $T[b_i, h]$需要 $(p-1)ql$ 次加法,计算所有 T_{b_i}需要 $(q-1)l$ 次比较,计算 T^B 需要 $l-1$ 次加法;所以,计算 T^B 总共需要 $(2q-1)pql+(p-1)ql+(q-1)l+l-1=2pq^2l-1$ 次域运算.

类似的讨论将说明算法 2 中变量 $T^C=\sum_{i=0}^{l-1}T_{c_i}$ 正确给出了迫使 $C_1=\cdots=C_p$ 的总代价,并且计算 T^C 总共需要 $2pq^2l-1$ 次域运算.

假设 $\min\{T^B, T^C\}\le K$. 当 $T^B\le T^C$ 时,不妨 $T[b_i, d]=T_{b_i}$,则我们要将所有 $b_{i+(j-1)l}(j=1,2,\cdots,p)$ 改变成 d,因为这样可以付出较低的代价迫使 $b_{i+(j-1)l}$都相等. 注意到在这一步结束时有 $a\leftarrow\left(\sum_{i=1}^{p}(-1)^{i+1}A_i, \sum_{i=1}^{p}(-1)^{i+1}A_{i+1}\right)$,如果我们要在第 $m+1$ 步改变 a_i 的值(在第 m 步 b_i 已被赋值为 d,相应的 a_i 和 a_{i+pl}已分别被赋值为 $d_{i,b}$和 $d_{i+pl,b}$)为 a_i+h,为了保持在第 m 步 $b_{i+(j-1)l}, j=1,2,\cdots,p$,都相等,我们必须要求

$$b_i=a_{i+(j-1)l}+h_{i,j-1}+a_{i+(j+p-1)l}+h_{i,j+p-1},\quad i=0,1,2,\cdots,l-1$$

这将多付出代价

$$\min_{\sum_{j=1}^{p}(-1)^{j+1}h_{i,j-1}=h}\Big\{\sum_{j=1}^{p}\mathrm{cost}[a_{i+(j-1)l}, a_{i+(j-1)l}+h_{i,j-1}]\Big|\ b_i=a_{i+(j-1)l}+h_{i,j-1}+a_{i+(j+p-1)l}+h_{i,j+p-1},\ j=1,2,\cdots,p\Big\}-\sum_{j=1}^{p}\mathrm{cost}[a_{i+(j-1)l}, d_{i+(j-1)l,b}]$$

在这种情况下,算法 2 正确计算了 $\mathrm{cost}[a_i, h]$. 这里计算 $\mathrm{cost}[a_i, h]$需要最多 $p^p(p-1)$ 次加法和最多 p^p-1 次比较,因而计算所有 $\mathrm{cost}[a_i, h]$最多需要 $(p^p(p-1)+p^p-1)2lq=2p^{p+1}ql-2ql$ 次域运算.

类似的讨论可以说明当 $T^B>T^C$ 时,算法 2 正确计算了 $\mathrm{cost}[a_i, a_i+h]$为

$$\min_{\sum_{j=1}^{p}h_{i,j-1}=h}\Big\{\sum_{j=1}^{p}\mathrm{cost}[a_{i+(j-1)l}, a_{i+(j-1)l}+h_{i,j-1}]\Big|\ c_i=a_{i+(j-1)l}+h_{i,j-1}-a_{i+(j+p-1)l}-h_{i,j+p-1},\ j=1,2,\cdots,p\Big\}-\sum_{j=1}^{p}\mathrm{cost}[a_{i+(j-1)l}, d_{i+(j-1)l,c}]$$

并且计算所有 $\mathrm{cost}[a_i, h]$最多需要 $2p^{p+1}ql-2ql$ 次域运算.

当 $\min\{T^B, T^C\}>K$ 时,我们无法迫使 $B_1=\cdots=B_p$ 或者 $C_1=\cdots=C_p$. 只好做赋值 $a\leftarrow\left(\sum_{i=1}^{p}A_{2i-1}, \sum_{i=1}^{p}A_{2i}\right)$,以便进行第 $m+1$ 步运算,如果我们要在第 $m+1$ 步改变 a_i 为 a_i+h,则在第 m 步需要改变 a_{i+2jl}为 $a_{i+2jl}+d_j$, $j=0,1,\cdots,p-1$,这里 $\sum_{j=0}^{p-1}d_j=h$. 所以,在第 $m+1$ 步改变 a_i 为 a_i+h 的代价 cost

$[a_i, a_i + h]$是在第 m 步改变 a_{i+2jl}为 $a_{i+2jl} + d_j$ 的代价和中的最小值,即

$$\min_{\sum_{j=0}^{p-1} d_j = h} \left\{ \sum_{j=0}^{p-1} \mathrm{cost}[a_{i+2jl}, a_{i+2jl} + d_j] \right\},$$

$$i = 0, 1, 2, \cdots, 2l - 1, h = 0, 1, 2, \cdots, q - 1.$$

因而在这种情况下,算法 2 正确计算了 $\mathrm{cost}[a_i, a_i + h]$. 类似上面的讨论知计算所有 $\mathrm{cost}[a_i, a_i + h]$最多需要 $2p^{p+1}ql - 2ql$ 次域运算.

现在我们已经证明了算法 2 并且在第 m 步用在计算改变元素所需代价的域运算次数最多为

$$(2lpq - 1) + 2(2pq^2 l - 1) + (2p^{p+1}ql - 2ql)$$
$$= 2qp^{n-m}(p + 2pq + p^{p+1} - 1) - 3$$

因此,算法 2 用在计算改变元素所需代价的域运算次数最多为

$$\sum_{m=1}^{n} [2qp^{n-m}(p + 2pq + p^{p+1} - 1) - 3]$$
$$= 2q(p + 2pq + p^{p+1} - 1)(p^n - 1)/(p - 1) - 3n.$$

这说明算法 2 用在计算改变元素所需代价的计算复杂度为 $O(N)$ $(N = 2p^n)$. 由于用在其它方面计算与算法 1 相同,而算法 1 的计算复杂度为 $O(N)$ $(N = 2p^n)$. 所以算法 2 的计算复杂度为 $O(N)$ $(N = 2p^n)$.

3 结论

本文提出 $GF(q)$ 上计算周期为 $2p^n$ 序列的 k-错线性复杂度的一个快速算法,它是我们在文献[9]中提出的确定周期为 $2p^n$ 周期序列的线性复杂度算法的推广. 该算法在 n 步内就可以计算出相应序列的 k-错线性复杂度. 我们也给出了算法的证明,并说明算法的计算复杂度为 $O(N)$ $(N = 2p^n)$.

参考文献:

[1] Ding C, Xiao G, Shan W. The Stability Theory of Stream Ciphers[M]. Lecture Notes in Computer Science Vol. 561. Berlin/ Heidelberg, Germany: Springer-Verlag, 1991.

[2] 冯登国,肖国镇. 序列周期稳定性新度量指标[J]. 电子学报, 1994, 22(1): 86 - 90.

[3] 陈克非. 序列的 d 复杂度[J]. 电子学报, 1989, 17(1): 112 - 113.

[4] Stamp M, Martin C F. An algorithm for k-error linear complexity of binary sequences with period 2^n[J]. IEEE Trans on Inform. Theory, 1993, 39(4): 1398 - 1401.

[5] Games R A, Chan A H. A fast algorithm for determining the complexity pseudo-random sequence with period 2^n[J]. IEEE Trans on Information Theory, 29(1): 144 - 146.

[6] Kaida T, Uehara S, Imamura K. An algorithm for the k-error linear complexity of sequences over GF(p^m) with period p^n, p a prime[J]. Information and Computation, 1999, 151(1): 134 - 147.

[7] Wei S, Chen Z, Xiao G. A fast algorithm for k-error linear complexity of a binary sequence[A]. Zhong Y X, Zhao Q. 2001 International Conferences on Info-tech and Info-net Proceedings[C]. IEEE Press, 2001. No Conference E, 152-157.

[8] Wei S, Xiao G, Chen Z. An efficient algorithm for k-error linear complexity[J]. Chinese Journal of Electronics, 2002, 11(2): 265 - 267.

[9] Wei S, Xiao G, Chen Z. A fast algorithm for determining the minimal polynomial of a sequence with period $2p^n$ over GF(q)[J]. IEEE Trans on Information Theory, 2002, 48(10): 2754 - 2758.

作者简介:

魏仕民 男,1962 年生于安徽省巢湖市,教授,1986 年和 1993 年分别在淮北煤炭师范学院和西北大学获得基础数学学士和硕士学位,2001 年在西安电子科技大学获密码学博士学位,2001 年 4 月至 2003 年 7 月,在北京大学信息科学技术学院软件研究所从事博士后研究,目前研究领域为应用数学、网络与信息安全. Email: weism02@yahoo.com.cn.