寻找布尔函数的零化子

谢 佳,王天择
(中国科学院软件研究所信息安全国家重点实验室,北京 100190)

摘 要: 通过解方程组来研究密码系统,是代数攻击的研究内容代.对方程组降次是降低求解复杂度的一种重要方法.为了达到这个目的,引入了布尔函数零化子的概念.然而迄今为止,尚未有求解零化子的有效算法.这篇文章提出了一种计算给定布尔函数的零化子集的算法.由前两个算法,可以得到给定布尔函数的零化子集的一组基;从第三个算法,可以得到最低次数的零化子.算法的复杂度与函数的单项式个数相关.对流密码来说,在很多情况下,相比以前的算法而言,这种算法的复杂度大为降低.最后,我们将给出一个实例,说明算法是如何工作的.

关键词: 代数攻击; 零化子; 代数免疫阶
中图分类号: TN918.1 **文献标识码:** A **文章编号:** 0372-2112 (2010) 11-2686-05

Finding the Annihilators of a Boolean Function

XIE Jia, WANG Tian-ze
(*State Key Lab of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing* 100190, *China*)

Abstract: Algebraic attack is used to study cryptosystems by solving equations. To lower the degrees of the equations is an important method to reduce the complexity to solve them. Annihilators is introduced to reach this aim, but there is not an efficient way to find the annihilators of a boolean function up to now. The article presents a kind of algorithm to work out the space of the annihilators of a given boolean function. We can find a basis of the annihilators set by the former two algorithms. Using the third algorithm, we can find the annihilators with lowest degree. The complexity is concerned with the number of monomials in the boolean function, and it is greatly reduced than that of the algorithms before in most cases for stream ciphers. We will present a toy example at the end of the article to show how the third algorithm works.

Key words: algebraic attack; annihilator; algebraic immunity

1 引言

2003 年,Nicolas T. Courtois 与 Wili. Meier 提出了针对流密码的代数攻击.代数攻击的基本原则,最早可以追溯到 Shannon 的思想[4]:这些方法的关键在于将整个算法用一个多变量代数方程组来描述,通过解方程组而获取密钥.影响方程组求解复杂度的一个重要参数是方程组的次数.通过这种方法,密码研究人员取得了一定成果.比如,在文献[1,2]中,提出了针对 Toyocrypt 加密算法的攻击;在文献[1,3]中,提出了针对 LiLi-128 加密算法的攻击.而这两种算法,可以很好地抵抗以前所有的已知攻击.

对流密码而言,当反馈移位寄存器为线性时,加密过程可以用式(1)表示.其中,$y_0, y_1, \cdots, y_{t-1}$为加密所需的密钥流.

由式(1)可见,此时密钥比特 y_i 是一个关于寄存器初始状态$(x_1, x_2, \cdots, x_n)$,次数为过滤函数 f 次数的函数.在流密码的设计时,f 一般具有较高的次数,因而最后得到的方程组的次数也比较高,这使方程组求解具有较高的复杂度.现在已经找到一些方法,来对方程组降次,因此密码学家重新考虑布尔函数的特性,使构造代数攻击免疫的布尔函数,成为密码学新的研究热点,并且现在已提出了一些构造方法,如文献[5,6]中各提出了一种构造达到最大代数免役阶的布尔函数的方法.

$$\begin{cases} f(x_1, x_2, \cdots, x_n) = y_0 \\ f(L(x_1, x_2, \cdots, x_n) = y_1 \\ \cdots \\ f(L^{t-1}(x_1, x_2, \cdots, x_n)) = y_{t-1} \end{cases} \tag{1}$$

为了对方程组降次,引入了布尔函数零化子的概念.如何利用零化子对方程组降次,在文献[13]中有较为详细的阐述.然而,现有的求零化子的算法并不实用.本文提出另一种算法,对流密码而言,这种算法在很多情况下可以更为有效的求出过滤函数的零化子.

2 预备知识

假定 B_n 为 n 元布尔函数集.在下文中,对任何 $n \geqslant 1$,我们将 B_n 视为 $GF(2)$ 上的 2^n 维向量空间.对任意 $f \in B_n$,在 $GF(2)$ 上,f 有唯一的多项式表示:$f(x_1, \cdots,$

收稿日期:2009-07-16;修回日期:2010-02-01
基金项目:国家 863 高技术研究发展计划(No.2007AA01Z447)

$x_n) = \sum a_\alpha m_\alpha$,其中,$\alpha \in \{0,1\}^n$.这被称为 f 的代数标准型(Algebraic Normal Form,简称 ANF).其中,对任意 $\alpha \in \{0,1\}^n$,$a_\alpha \in GF(2)$,$m_\alpha = \prod_{i,a_i=1} x_i$. α_i 为 α 的第 i 位,对 $\alpha \in \{0,1\}^n$,定义 α 的汉明重量 $wt(\alpha) = |\alpha| = |\{i \mid \alpha_i = 1\}|$.布尔函数 f 的次数 $\deg(f) = \max\{wt(\alpha), a_\alpha = 1\}$.

在对流密码的代数攻击中,我们需要对方程组降次以降低求解的复杂度.因此,引入了布尔函数的零化子和代数免疫阶(Algebraic immunity,简称 *AI*)的概念.对布尔函数 f,若非零函数 g 满足 $f \cdot g = 0$,则称 g 是 f 的零化子.

定义 1　设 $f \in B_n$,f 的零化子集:

$$An(f) = \{g \in B_n \mid f \cdot g = 0\};$$

$$An_d(f) = \{g \mid g \in An(f), \deg(g) \leqslant d\}.$$

显然,$An_d(f)$ 是 B_n 的一个子空间,因此,我们可以用一组基来描述零化子集.

定义 2　设 $f \in B_n$,$\mathrm{supp}(f) = \{x \in \{0,1\}^n \mid f(x) = 1\}$.

设 $f \in B_n$,若 $\mathrm{supp}(f) = 2^{n-1}$,则称 f 是平衡的.

由前述定义可得,$AI(f) = \min\{\deg(g) \mid g \in An(f) \cup An(f+1)\}$.

下面给出几个有关代数免役阶的结论.

引理 1[13]　设 $f \in B_n$,则 $AI(f) \leqslant \lceil n/2 \rceil$.

引理 2[7]　设 f 为任意 n 元平衡布尔函数,当 $n \to \infty$,$\Pr\{AI(f) \leqslant \mu n\} \to 0$,$\mu \approx 0.22$.

由引理 1 和引理 2,除少数情况外,当 n 充分大时,我们大致可以确定布尔函数代数免疫阶的上界与下界.

引理 3[7]　设 $f \in B_n$,$An(f) = <1 + f>$.

计算 n 元布尔函数 $f(x)$ 的代数免疫阶的最初设想[6],是利用 $\mathrm{supp}(f)$ 中的向量构造一个矩阵,通过对矩阵高斯消去以得到函数的代数免疫阶.若 $f(x)$ 的代数免疫阶为 d,算法的复杂度为 $O(n^{2.37d})$.

文献[7]提到了一种降低矩阵规模的方法,这样,可以降低高斯消去的复杂度.这种方法的思想是,由 $\mathrm{supp}(f)$ 中汉明重量较小的元素,可以得到较为简单的方程,通过替换矩阵中相应的行,达到缩小矩阵规模的目的.然而,因为缺少模拟结果,很难估计这种方法在实际中替换步骤的时间复杂度.

由于上述两种算法的时间复杂度很高,密码学家们尝试寻找更为有效地算法.文献[9]提出了一种寻找对称布尔函数零化子的有效算法,不过对称布尔函数在密码学上用途并不广泛.此外,文献[10]提出一种利用 Grobner 基来寻找零化子的算法,但算法效率也不是很高.文献[11]提出一种算法,可以有效地用于寻找快速代数攻击与代数攻击之间的关系.文[11]的算法中,对求解零化子的矩阵三角化过程进行了优化,寻找 n 变元布尔函数的 d 次零化子的时间复杂度为 $O(\binom{n}{d}^2)$.文献[12]提出了一种时间复杂度为 $O(n^d)$ 的概率性算法.这种算法将函数递归地分拆为子函数,通过对子函数的零化子的求解,从而得到原布尔函数的零化子.

以前的大部分算法都有一个共同的缺陷.它们构造了矩阵,并且通过矩阵操作来得到零化子,这导致了很高的时间和空间复杂度.由引理 2,当 n 充分大时,在绝大多数情况下,$AI(f) \geqslant 0.22n$,即使是效率最高的文[12]中的算法,$O(n^d)$ 的复杂度也超过了穷举.因此,寻找高效率的确定布尔函数零化子的算法,仍然是一个开放问题.

由文献[1],如果能得到过滤函数的多个零化子,我们将可以从一个密钥比特得到多个方程,这样将大为降低恢复 LFSR 初始状态所需的密钥比特数.

3　本文方法

本文从另一个角度来考虑这个问题.这里,我们试图寻找一种方法,可以求解出给定布尔函数的零化子集合,进而确定具有最低次数的零化子.

首先,我们引入布尔函数的类 DNF 范式表示.对应于 $\neg f_1$ 的函数为 $f_1 + 1$;对应于 $f_1 \wedge f_2$ 的函数为 $f_1 \cdot f_2$;对应于 $f_1 \vee f_2$ 的函数为 $(f_1 + 1)(f_2 + 1) + 1$.

设 A,B 为两个集合,集合 $A + B$ 定义为:$A + B = \{a + b \mid a \in A, b \in B\}$.

我们先给出两个定理.

定理 1　设 $f_1, f_2 \in B_n$,有结论

①$An(f_1) \cap An(f_2) = An(f_1 \vee f_2)$ ②$An(f_1 + 1) \cap An(f_2 + 1) = An(f_1 \cdot f_2 + 1)$ ③$An(f_1) + An(f_2) = An(f_1 \wedge f_2)$

证明

①若 $f \in An(f_1 \vee f_2)$,即 $f \in An((f_1 + 1)(f_2 + 1) + 1)$,因此 $f \cdot ((f_1 + 1)(f_2 + 1) + 1) = 0$.由引理 3,$f$ 在 $(f_1 + 1)(f_2 + 1)$ 生成的理想中.设 $f = (f_1 + 1)(f_2 + 1) \cdot f'$,显然,有结论:$f_1 \cdot (f_1 + 1)(f_2 + 1) \cdot f' = f_2 \cdot (f_1 + 1) \cdot (f_2 + 1) \cdot f' = 0$.因此,$f \in An(f_1) \cap An(f_2)$.

由上可得 $An(f_1 \vee f_2) \subseteq An(f_1) \cap An(f_2)$.

在另一方面,若 $f \in An(f_1) \cap An(f_2)$,$f \cdot f_1 = f \cdot f_2 = 0$.由于 $f \cdot ((f_1 + 1) \cdot (f_2 + 1) + 1) = f(f_1 f_2 + f_1 + f_2) = 0$,$f \in An((f_1 + 1)(f_2 + 1) + 1)$,即 $f \in An(f_1 \vee f_2)$.

因此,$An(f_1) \cap An(f_2) \subseteq An(f_1 \vee f_2)$.

综上,$An(f_1) \cap An(f_2) = An(f_1 \vee f_2)$.　**证毕.**

②由①可得,

$$An(f_1 + 1) \cap An(f_2 + 1) = An((f_1 + 1) \vee (f_2 + 1))$$
$$= An(((f_1 + 1) + 1)((f_2 + 1) + 1) + 1)$$

$= An(f_1 \cdot f_2 + 1)$. **证毕.**

③显然 $An(f_1) + An(f_2) \subseteq An(f_1 \wedge f_2)$. 在另一方面,由引理 3,若 $f \in An(f_1 \wedge f_2)$, $f = (f_1 \cdot f_2 + 1) \cdot f' = (f_1 + 1) \cdot f_2 \cdot f' + (f_2 + 1) \cdot f'$,并且 $(f_1 + 1) \cdot f_2 \cdot f' \in An(f_1)$, $(f_2 + 1) \cdot f' \in An(f_2)$. 因此 $f \in An(f_1) + An(f_2)$. 有结论 $An(f_1) + An(f_2) \supseteq An(f_1 \wedge f_2)$. 也就是说, $An(f_1) + An(f_2) = An(f_1 \wedge f_2)$. **证毕.**

进一步地,我们有以下结论.

定理 2 设 f_i 为单项式, $f_i = x_{i_1} x_{i_2}, \cdots, x_{i_k}$,则 $An(f_i + 1) = < x_{i_1} x_{i_2}, \cdots, x_{i_k} >$; $An(f_i) = < x_{i_1} + 1, x_{i_2} + 1, \cdots, x_{i_k} + 1 >$.

证明 在这里,当理想 I 的生成元是一组单项式 $f_1, \cdots, f_m$ 时,我们称 I 是由这组单项式生成的理想: $I = \{f_i g_i \mid g_i \in B_n\}$.

由引理 2 可得, $An(f_i + 1) = <f_i> = < x_{i_1} x_{i_2}, \cdots, x_{i_k} >$.

由定理 1 中的③, $An(f_i) = An(x_{i_1} x_{i_2}, \cdots, x_{i_k}) = An(x_{i_1}, \cdots, x_{i_t}) + An(x_{i_{t+1}}, \cdots, x_{i_k})$.

重复上述分拆过程,得:

$$\begin{aligned} An(f_i) &= An(x_{i_1}) + An(x_{i_2}) + \cdots + An(x_{i_k}) \\ &= < x_{i_1} + 1 > + < x_{i_2} + 1 > + \cdots + < x_{i_k} + 1 > \\ &= < x_{i_1} + 1, x_{i_2} + 1, \cdots, x_{i_k} + 1 >. \end{aligned}$$ 证毕.

设 $f_i = x^{\alpha_i}$, f 为含有 m 个单项式的布尔函数,且 $f = \sum_{i=1}^{m} f_i$.

为了得到给定布尔函数的零化子,我们需要先将函数的代数标准型(ANF)转换为类 DNF 范式.

对函数 $f_1 + f_2$,有:

$f_1 + f_2 = (\neg f_1 \wedge f_2) \vee (f_1 \wedge \neg f_2)$;

$f_1 + f_2 + 1 = \neg (f_1 + f_2) = (\neg f_1 \wedge \neg f_2) \vee (f_1 \wedge f_2)$.

由定理 1,我们可以得到 $An(f_1 + f_2)$, $An(f_1 + f_2 + 1)$. 如此递推,可以得到 $An(\sum_{i=1}^{m} f_i)$ 与 $An(\sum_{i=1}^{m} f_i + 1)$. 因此,我们可得以下算法.

算法 1(寻找 f 与 $f+1$ 的零化子集, $f = \sum_{i=1}^{m} f_i$)

输入: $f = \sum_{i=1}^{m} f_i$;

输出: $An(f)$, $An(f+1)$.

$f' = f_1$;

for $i = 2$ to m do

$F = f' + f_i = (f' \wedge \neg f_i) \vee (\neg f' \wedge f_i)$;

$\neg F = f' + f_i + 1 = (\neg f' \wedge \neg f_i) \vee (f' \wedge f_i)$;

$f' = F$;

由定理 1,计算 $An(f')$, $An(\neg f')$;

end

return $An(f)$, $An(\neg f')$

本文的算法结果里,均是返回集合的一组基来表示该集合.

设 $f_1 = x_{i_1}, \cdots, x_{i_n}$, $f_2 = x_{j_1}, \cdots, x_{j_m}$.

现在我们讨论集合 $An((f_1 \wedge \neg f_2) \vee (\neg f_1 \wedge f_2))$ (即 $An(f_1 + f_2)$)中元素的形式.

$An((f_1 \wedge \neg f_2) \vee (\neg f_1 \wedge f_2)) = < 1 + x_{i_1}, \cdots, 1 + x_{i_n}, x_{j_1}, \cdots, x_{j_m} > \cap < 1 + x_{j_1}, \cdots, 1 + x_{j_m}, x_{i_1}, \cdots, x_{i_n} >$.

令理想 $A = < f_1, \cdots, f_{n+1} > = < 1 + x_{i_1}, \cdots, 1 + x_{i_n}, x_{j_1} \cdots x_{j_m} >$, $B = < g_1, \cdots, g_{m+1} > = < 1 + x_{j_1}, \cdots, 1 + x_{j_m}, x_{i_1} \cdots x_{i_n} >$. 由于 $x_{j_1} \cdots x_{j_m} + (1 + x_{j_1}) \cdot x_{j_2} \cdots x_{j_m} + (1 + x_{j_2}) \cdot x_{j_3} \cdots x_{j_m} + \cdots (1 + x_{j_{m-1}}) \cdot x_{j_m} + (1 + x_{j_m}) = 1$,

因此理想 A 和理想 B 互素,有结论 $A \cap B = \{\sum k_{ij} f_i g_j \mid 1 \leqslant i \leqslant n+1, 1 \leqslant j \leqslant m+1\}$,这里, k_{ij} 为某布尔函数.

类似的,我们可以得到 $An(f_1 + f_2 + 1)$. 具体如下: $An((\neg f_1 \wedge \neg f_2) \vee (f_1 \wedge f_2)) = < x_{i_1} \cdots x_{i_n}, x_{j_1} \cdots x_{j_m} > \cap < 1 + x_{i_1}, \cdots, 1 + x_{i_n}, 1 + x_{j_1}, \cdots, 1 + x_{j_m} >$.

令 $A' = <f'_1, f'_2> = < x_{i_1} \cdots x_{i_n}, x_{j_1} \cdots x_{j_m} >$, $B' = < g'_1, \cdots, g'_{m+n} > = < 1 + x_{i_1}, \cdots, 1 + x_{i_n}, 1 + x_{j_1}, \cdots, 1 + x_{j_m} >$.

同样地, A', B' 互素,因此 $A' \cap B' = \{\sum t_{ij} f'_i g'_j \mid 1 \leqslant i \leqslant 2, 1 \leqslant j \leqslant m+n\}$, t_{ij} 为某布尔函数.

若 $A = <f_1, \cdots, f_n>$, $B = <g_1, \cdots, g_m>$, $f_i, g_j \in B_n$,显然: $A \cap B \supseteq \{\sum k_{ij} T_{ij} \mid 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m\}$ (2)

这里, T_{ij} 为 f_i 和 g_j 的最小公倍式.

我们未能给出一般情形下完整的证明. 然而,在执行算法 1 时,利用式(2),我们最少可以得到所求零化子集的某个子集.

下面考虑算法 1 的效率问题.

为方便起见,不妨设当 $i_1 < i_2$, $\deg(f_{i_1}) \geqslant \deg(f_{i_2})$, $\deg(f_i) = d_i$.

算法 1 的第一轮中,集合 $An(f_1 + f_2)$ 的基有 $O(d_1 d_2)$项;同时, $An(\neg (f_1 + f_2))$的基有 $O(d_1 + d_2)$项. 确定 $An(f_1 + f_2)$的基的时间复杂度为 $O(d_1 d_2)$;确定 $An(\neg (f_1 + f_2))$的基的时间复杂度为 $O(d_1 + d_2)$. 然而,在接下来的循环里,确定返回零化子集合的基的时间复杂度,将呈指数增长. 因此,我们需要对算法 1 进行优化.

$$f_1 + f_2 + f_3 + f_4 = (\neg (f_1 + f_2) \wedge (f_3 + f_4)) \vee ((f_1 + f_2) \wedge \neg (f_3 + f_4));$$

$$f_1 + f_2 + f_3 + f_4 + 1 = (\neg (f_1 + f_2) \wedge \neg (f_3 + f_4)) \vee ((f_1 + f_2) \wedge (f_3 + f_4)).$$

在空间 $An((\neg (f_1 + f_2) \wedge (f_3 + f_4))$中,基有 $O(d_1 + d_2 + d_3 d_4)$项;在空间 $An((f_1 + f_2) \wedge \neg (f_3 + f_4))$中,

基有 $d_1d_2+d_3+d_4$ 项.由(2)可得,要计算 $An(f_1+f_2+f_3+f_4)$的时间复杂度为 $O((d_1+d_2+d_3d_4)(d_1d_2+d_3+d_4)=\max(O(d_1d_2d_3d_4),O(d_1^2d_2))$;同理,计算 $An(f_1+f_2+f_3+f_4+1)$的时间复杂度为 $O((d_1+d_2+d_3+d_4)(d_1d_2+d_3d_4))=O(d_1^2d_2)$.

类似的,确定 $An(f_1+f_2+f_3+f_4+f_5+f_6+f_7+f_8)$所耗费的时间复杂度为 $\max(O(d_1d_2d_3d_4d_5d_6d_7d_8),O(d_1^3d_2^2d_3d_4))$;确定 $An(f_1+f_2+f_3+f_4+f_5+f_6+f_7+f_8+1)$耗费的时间复杂度为:$O(d_1^3d_2^2d_3d_4)$.

如此,可将结论递推至 $f=\sum_{i=1}^{2^k}f_i$.

确定 $An(f)$的时间复杂度为 $\max(O(\prod_{i=1}^{2^k}d_i),O(\prod_{i=1}^{2^k}d_i^{k-\lceil\log_2 i\rceil}))$;确定 $An(\neg f)$的时间复杂度为 $O(\prod_{i=1}^{2^k}d_i^{k-\lceil\log_2 i\rceil})$.

因此,算法的复杂度为 $\max(O(\prod_{i=1}^{2^k}d_i),O(\prod_{i=1}^{2^k}d_i^{k-\lceil\log_2 i\rceil}))$.

由上述讨论可知,若在递推的过程中,将子函数两两结合以求解零化子集,算法的复杂度将不再随着轮数增加而呈指数增长.我们通过这种方法,对算法 1 进行优化.

设 $m=2^{k_1}+2^{k_2}+\cdots+2^{k_p}$(当 $i<j,k_i>k_j$),由前面的分析可知,确定 $An(f+1)$的复杂度为:

$$O(\prod_{j=1}^{p}\prod_{i=1}^{2^{k_j}}d_{2^{k_1}+\cdots+2^{k_{j-1}}+i}^{k_j-\lceil\log_2 i\rceil}).$$

从上面的分析可见,算法 2 的时间复杂度为 $\max(O(\prod_{i=1}^{m}d_i),O(\prod_{j=1}^{p}\prod_{i=1}^{2^{k_j}}d_{2^{k_1}+\cdots+2^{k_{j-1}}+i}^{k_j-\lceil\log_2 i\rceil}))$,其中 d_i 为 f 中第 i 个单项式所含的变元个数.

算法 2(寻找 f 与 $f+1$ 的零化子,$f=\sum_{i=1}^{m}f_i$)

输入:$f=\sum_{i=1}^{m}f_i$;
输出:$An(f),An(f+1)$.
$A=\{f_1,\cdots,f_m\},B=\phi$
L1　while $|A|\geqslant 2$ do
　$F_{ij}=f_i+f_j,\ f_i,f_j\in A$;
　计算 $An(F_{ij})$, $An(\neg F_{ij})$;
　$B=B\cup\{F_{ij}\}$, $A=A-\{f_i,f_j\}$;
　end while;
　$A=A\cup B,B=\phi$;
　if $|A|=1$ break;
　goto *L1*;
　return $An(f'),An(\neg f'),f'\in A$

由上述算法,我们可以得到布尔函数的零化子集(或者某子集).但是在实际中,我们更关心那些达到最低次数的零化子.

显然,如果 $f_1'\in An(f_1),f_2'\in An(f_2)$,则

$$f_1'\cdot f_2'\in An(f_1+f_2)\tag{3}$$

为了提高生成密钥的效率,过滤函数的绝大部分单项式的次数并不高.由式(3),我们通过考虑过滤函数的高次部分,来确定函数的具有最低次数的零化子.

在下面的算法里,我们以单项式的次数进行分类,相同次数的单项式组成原函数的子函数.从次数较高的子函数开始,若算法运行使已求得的最低零化子次数降低,则算法继续;否则,说明我们已经求得了最低次数的零化子.

设 $f=F_{i_1}+F_{i_2}+\cdots+F_{i_t}$,$F_{i_k}$ 中的单项式的次数均为 $i_k(1\leqslant i_k\leqslant n,k=1,2,\cdots,t)$.

算法 3(寻找 f 与 $f+1$ 的最低次数的零化子,$f=\sum_{i=1}^{m}f_i$)

输入:$f=F_{i_1}+F_{i_2}+\cdots+F_{i_t}$,$F_{i_k}$ 仅包含次数为 i_k 的单项式 $1\leqslant i_k\leqslant n(j=1,2,\cdots,t)$;
输出:最低次数零化子集.
①$f=F_{i_t}$,对 f 执行算法 2,$An(f),An(\neg f)$为返回集合,并且 d_{i_t} 为集合 $An(f),An(\neg f)$中的最低次数;
　$AI(f)=d_{i_t}+i_{t-1}$;
②for $k=t-1$ downto 1 do
　$f=f+F_{i_k}$,对 f 执行算法 2,$An'(f),An'(\neg f)$ 为返回集合,并且 d_{i_k}为 $An'(f),An'(\neg f)$中的最低次数;
　if $d_{i_k}+i_{k-1}<AI(f)$,then
　$An(f)=An'(f),An(\neg f)=An'(\neg f)$,
　$AI(f)=d_{i_k}+i_{k-1}$;
　else break;
return $An(f),An'(\neg f)$

算法所得的零化子为 $f'\cdot(F_{i_k}+F_{i_{k-1}}+\cdots+F_{i_1}+1)$,这里,$k$ 为算法 3 中断时的值,f' 为 $An(f),An'(\neg f)$中次数最低的函数.

4　一个实例

LiLi-128 使用的过滤函数为:

$$\begin{aligned}f=\ &x_2+x_3+x_4+x_5+x_6x_7+x_1x_8+x_2x_8+x_1x_9+x_3x_9+\\&x_4x_{10}+x_6x_{10}+x_3x_7x_9+x_4x_7x_9+x_6x_7x_9+x_3x_8x_9+\\&x_6x_8x_9+x_4x_7x_{10}+x_5x_7x_{10}+x_6x_7x_{10}+x_3x_8x_{10}+\\&x_4x_8x_{10}+x_2x_9x_{10}+x_3x_9x_{10}+x_4x_9x_{10}+x_5x_9x_{10}+\\&x_3x_7x_8x_{10}+x_5x_7x_8x_{10}+x_2x_7x_9x_{10}+x_4x_7x_9x_{10}+\\&x_6x_7x_9x_{10}+x_1x_8x_9x_{10}+x_3x_8x_9x_{10}+x_4x_8x_9x_{10}+\\&x_6x_8x_9x_{10}+x_4x_6x_7x_9+x_5x_6x_7x_9+x_2x_7x_8x_9+\\&x_4x_7x_8x_9+x_4x_6x_7x_9x_{10}+x_5x_6x_7x_9x_{10}+x_3x_7x_8x_9x_{10}\end{aligned}$$

$+ x_4x_7x_8x_9x_{10} + x_4x_6x_7x_8x_9 + x_5x_6x_7x_8x_9 + x_4x_6x_7x_8x_9x_{10} + x_5x_6x_7x_8x_9x_{10}$;算法在 $k=2$ 时终止.

我们可以确定 f 的一个零化子:

$(1+x_2+x_3+x_4+x_5+x_6x_7+x_1x_8+x_2x_8+x_1x_9+x_3x_9+x_4x_{10}+x_6x_{10})(1+x_9)(1+x_{10})$.(这个函数等价于$(1+x_2+x_3+x_4+x_5+x_6x_7+x_1x_8+x_2x_8)(1+x_9)(1+x_{10})$.)

过滤函数中单项式的次数较为接近,确定 $An(F_3+F_4+F_5+F_6)$的时间复杂度为 $O(6^2\cdot5^6\cdot4^{13}\cdot3^4)\approx O(2^{67})$.

由原来的过滤函数建立的方程组次数为 6,求解的复杂度为 $O((C_{128}^6)^{2.37})\approx2^{99}$.上述算法寻找到的零化子的次数为 4,求解所建立的方程组的复杂度为 $O((C_{128}^4)^3)\approx2^{66}$.此外,所需要的密钥流由 C_{128}^6降为 C_{128}^4.

5 结论

我们给出了三个算法,以确定给定布尔函数零化子集的一组基,算法的复杂度与函数中单项式的个数相关.算法 1 的复杂度过高,对其进行优化后得到了算法 2;另外,前面两个算法,得到的零化子集合较为复杂,为了得到具有最低次数的零化子,我们给出了算法 3.最后,我们给出了一个实例,说明算法 3 是如何工作的.此外,我们可以由这些算法得到多个零化子,因而,确定 LFSR 初始状态所需的密钥比特数将大为降低.

参考文献:

[1] Nicolas T Courtois, Wili. Meier. Algebraic attacks on stream ciphers with linear feedback[A]. Adcances in Cryptology-EUROCRYPT 2003, LNCS 2656[C]. Springer-Verlag, 2003. 346 – 359.

[2] M Mihaljevie, H Imai. Cryptanalysis of Toyocrypt-HIS stream cipher[J]. IEICE Transactions on Fundamentals, 2002, E85-A: 66 – 73.

[3] Steven Babbage, Cryptanalysis of LILI-128[R]. 2001. http://www.cosic.esat.kuleuven.ac.be/nessie/reports/.

[4] C E Shannon. Communication theory of secrecy system[J]. Bell System Technical Journal, 1949, 28(4): 656 – 715. http://netlab.cs.ucla.edu/wiki/files/shannon1949.pdf.

[5] D K Dalai, S Maitra, S Sarkar. Basic theory in construction of boolean functions with maximum possible annihilator immunity [J]. Designs, Codes and Cryptography 2006, 40(1): 41 – 58.

[6] Na Li, Wen-Feng, Qi. Construction and analysis of boolean functions of 2t + 1 variables with maximum algebraic immunity [A]. Asiacrypt 2006, LNCS 4284[C]. Springer-Verlag, 2006. 84 – 98.

[7] W Meier, E Pasalic, C Carlet. Algebraic attacks and decomposition of boolean functions[A]. In Advances in Cryptology-EUROCRYPT 2004, LNCS 3027[C]. Springer-Verlag, 2004. 474 – 491.

[8] Anne Canteaut. Open problems related to algebraic attacks on stream ciphers[A]. In International Workshop on Coding and Cryptography 2005, LNCS 3969[C]. Springer Verlag, 2006. 120 – 134.

[9] A. Braeken, B. Praneel. On the algebraic Immunity of symmetric boolean fucntions[A]. In Indocrypt 2005, LNCS 3797[C]. Springer Verlag, 2005. 35 – 48.

[10] G Ars, J Faugere. Algebraic attacks over GF(q)[A]. In Progress in Cryptology-Indocrypt 2004, LNCS 3348[C]. Springer-Verlag, 2004. 84 – 91.

[11] F Armknecht, C Carlet, P Gaborit, et al. Efficient computation of algebraic immunity for algebraic and fast algebraic attacks [A]. In Eurocrypt 2006, LNCS 4004[C]. Springer Verlag, 2006. 147 – 164.

[12] F Didier, J Tillich. Computing the algebraic immunity efficiently[A]. In FSE 2006. LNCS 4047[C]. Springer Verlag, 2006. 359 – 374.

[13] 张文英,武传坤,于静之.密码学中布尔函数的零化子[J].电子学报,2006,34(1):51 – 54.
Zhang Wen-ying, Wu Chuan-kun, Yu Jing-zhi. On the annihilators of cryptographic boolean functions[J]. Acta Electronica Sinica, 2006, 34(1): 51 – 54. (in Chinese)

作者简介:

谢 佳 男,1980 年生于湖南南县,本科毕业于北京航空航天大学,现为中科院软件所博士生,研究方向为密码学.
E-mail: xiejia@is.iscas.ac.cn

王天择 男,1983 年生于河南郑州,本科毕业于西安电子科技大学,现为中科院软件所博士生,研究方向为密码学.
E-mail: wtz83@is.iscas.ac.cn