第5章　代数分析方法

本章内容提要

相比于统计分析方法，代数分析是另一种对于基于线性反馈（如 LFSR）的序列密码的典型分析方法。虽然代数分析的思想可以追溯到 Shannon[1] 于 1949 年发表的经典论述，但直到 2003 年，Courtois 等[2] 才正式提出了序列密码代数分析的思想，并对基于线性反馈的序列密码进行了代数分析，将之应用于序列密码 Toyocrypt 和 LILI-128 的分析。对基于线性反馈的序列密码而言，由于线性关系无法增加代数次数，序列密码的内部状态和密钥流比特通过非线性滤波（也称过滤）函数或者组合函数直接组合，缺乏动态增长，易导致关于一定个数变量的大量固定次数代数关系式，从而易被线性化方法恢复出初始状态变量。如果非线性组合函数的代数次数比较低，或者通过倍乘等化简技巧可以转化成代数次数较低的情况，则线性反馈的初始状态可以通过线性化这些非线性方程所包含的单项式来恢复，这也是目前为止唯一可进行理论分析的代数分析方法。其他方法，如采用 Gröbner 基、XL、SAT 等，虽然只需较少的密钥流即可攻击成功，但缺乏对于序列密码本身构造的有效利用，只是现有解方程方法的直接使用，从而难以获得广泛的关注和成功使用。在 2003 年，Courtois[3] 又进一步改进了代数分析方法，在获得连续密钥流的情况下，提出了快速代数分析的思想，可以利用 B-M 算法等方法局部地降低代数方程的次数，从而减少单项式数目，降低复杂度。作为代数分析方法的一个进展，在 2009 年，Dinur 等[4] 提出了立方分析方法，其基本思想是：利用高阶差分的性质来压缩原迭代函数，在获得的压缩结果是线性方程或者低次方程的情况下，可以通过解方程获得初始密钥。随后，人们又提出了立方区分器、动态立方分析、条件差分分析等方法，并被应用于序列密码 Trivium、Grain-128 和 Grain-128a 的分析，其典型思想是：利用可控制的 IV 变量来零化某些深度轮数的某些位置变量，得到较易被区分的结果函数。

本章主要介绍代数分析、快速代数分析、改进的快速代数分析和立方分析 4 种代数分析方法。

本章重点

- 代数分析方法的基本原理。
- 快速代数分析方法的基本原理。
- 改进的快速代数分析方法的基本原理。
- 立方分析的基本思想。
- 代数免疫度的发展背景、基本概念和特征。

5.1　基于线性反馈的序列密码的代数分析方法

本节主要介绍基于线性反馈（如 LFSR）的序列密码（如滤波生成器、组合生成器）的代数分析方法。这类密码的安全性已受到人们的高度关注，如相关分析。相关分析的应用范

围也在逐渐拓展。例如,文献[5]构造了一个使用所有变量的非线性低次数函数,它与原函数具有相关性,或者说低次数非线性逼近,这种方法并不是新的,可参见文献[6]。然而,这种方法的应用没有得到充分的重视,可能是由于最近人们才意识到求解低次数非线性多变量方程组的有效算法的存在性[5,7-9]。文献[2]一般化了上述低次数非线性逼近方法,称之为代数分析方法。这种一般化的方法,即使没有好的低次数线性逼近,仍可应用于序列密码的分析,并可分析一大类序列密码。本节主要介绍文献[2]中的相关工作。

5.1.1 序列密码的代数分析方法

本节综述和扩展了文献[5]中的一般性策略,把序列密码的攻击归纳为求解一组多变量方程。

1. 可被攻击的序列密码模型

这里只考虑同步序列密码,每个状态独立于明文,从前一个状态产生。从原理上,我们也只考虑以已知方式规则钟控的序列密码。然而这个条件有时能被放宽,如下面将要介绍的对 LILI-128 的攻击。

为简单起见,将序列密码限制为二元序列密码,该密码的状态和密钥流由一系列比特组成,并在每个时刻生成一个比特。我们也仅讨论计算下一状态的"连接函数"在 F_2 上是线性的这种情况。把这个"连接函数"记为 L,假定 L 是公开的,只有状态是秘密的。我们也假定从状态计算输出比特的函数 f 是公开的,并不依赖于该序列密码的秘密密钥。函数 f 也被称为一个非线性滤波函数。这类序列密码包括滤波生成器和组合生成器。这类序列密码的分析问题可描述为如下的问题。

设$(k_0,k_1,\cdots,k_{n-1})$是初始状态,则这类序列密码的输出即密钥流可按如下方式给出:

$$\begin{cases} b_0 = f(k_0,k_1,\cdots,k_{n-1}) \\ b_1 = f(L(k_0,k_1,\cdots,k_{n-1})) \\ b_2 = f(L^2(k_0,k_1,\cdots,k_{n-1})) \\ \quad\vdots \end{cases}$$

我们的问题是从某一密钥流比特 $b_i(i=0,1,2,\cdots)$子集恢复密钥 $k=(k_0,k_1,\cdots,k_{n-1})$。将执行一个部分已知明文攻击,即已知明文的一些比特和对应的密文比特。这些比特不必是连续的。例如,如果明文用拉丁字母书写并且不使用太多的特殊字符,很可能所有的字符的最高比特都等于 0。如果文本是充分长的,这个对攻击者来说是足够的。这将是或几乎是一个唯密文攻击。在这里的攻击中,仅假定有密钥流 b_i 中已知位置的 m 个比特。

2. 攻击的基本思想

在时刻 t,当前的密钥流比特给出了一个等式:$f(s)=b_t$,s 是当前的状态。主要的新观点在于用 $g(s)$乘以 $f(s)$,即,通常 $f(s)$是高次数的,通过乘以一个良好选择的多变量多项式 $g(s)$,使得 $f(s)g(s)$具有较低的次数,记其次数为 d。然后,如果 $b_t=0$,就得到一个低次数方程 $f(s)g(s)=0$。反过来,这也给出了一个关于内部状态比特 $k_i(i=0,1,\cdots,n-1)$的低次数 d 的多变量方程。如果有充分多的密钥流比特且对每一比特都得到一个这样的方程,就能获得可被有效地求解的超定多变量方程组(也就是许多方程)。也就是说,代数分析方法可将一类序列密码的分析问题转化为低次数多变量方程组的求解问题。

3. f 的设计准则和已知攻击

设 f 是用于组合密码的线性部分的输出的布尔函数，如函数的输入是 LFSR 状态的一些比特。对这样的函数的通常要求是：f 首先是平衡的并具有高的代数次数；为了抵抗线性攻击和相关攻击，f 必须具有高的非线性度和高的相关免疫阶。

文献[5]中给出了关于 f 的一个额外的准则：f 也不能与低次数非线性多变量函数有很强的相关性，否则有效的攻击是可能的，如对 Toyocrypt 序列密码的攻击[5]。也就是说，如果 f 满足下列两种情形之一，就有可能存在有效的攻击。

情形 S1：布尔函数 f 具有一个低的代数次数 D（经典的准则）。

情形 S2：f 能用一个具有低次数的布尔函数以接近 1 的概率逼近（文献[5]中的新准则）。这个概率通常表示为 $1-\varepsilon$，ε 是某一较小的数。

文献[5]中使用情形 S2 对 Toyocrypt 进行了实际的攻击。

4. f 的新的假设和新的攻击情形

情形 S3：多变量多项式 f 有某一低次数（设其次数为 d）倍式 fg，g 是某一非零多变量多项式。

情形 S4：f 有某一倍式 fg，使得 fg 能用一个具有低次数的函数以接近 1 即（$1-\varepsilon$）的概率逼近，ε 是某一较小的数。

更重要的问题是如何使用低次数倍式，如果情形 S3 和 S4 都可利用，则说明新的攻击不必要求 f 具有一个低的代数次数或其良好逼近具有一个低的次数。

在情形 S1 和 S2 中，对每个在位置 t 的已知密钥流比特 b_t，获得一个具体的值 $b_t=f(s)$，并且这样就可得到方程 $b_t=f(L^t(k_0,k_1,\cdots,k_{n-1}))$。对此，$f$ 不得不具有低次数。

在情形 S3 和 S4 中，对每个在位置 t 的已知密钥流比特 $b_t=f(s)$，得到

$$f(s)g(s)=b_tg(s)$$

并且因为状态在时刻 t 是 $s=L^t(k_0,k_1,\cdots,k_{n-1})$，归结起来是

$$f(L^t(k_0,k_1,\cdots k_{n-1}))g(L^t(k_0,k_1,\cdots,k_{n-1}))=b_tg(L^t(k_0,k_1,\cdots,k_{n-1}))$$

对每个密钥流比特得到一个多变量方程。这个方程具有很低的次数，无须 f 具有低次数，也无须 f 有一个低次数逼近。在情形 S3 下，攻击的基本形式也要求 g 具有低次数，这一点可通过将情形 S3 分成以下 4 种情形来说明。

情形 S3a：存在 g，其次数比 f 的次数低，使得

$$f(s)g(s)=b_tg(s)=0$$

当 $b_t=1$ 时，一定有 $g(s)=0$。因为 g 的次数比 f 的次数低，这样就能得到一组方程 $g(s)=0$ 并且更易求解。

情形 S3b：存在 g，其次数比 f 的次数低，使得

$$(1\oplus f(s))g(s)=(1\oplus b_t)g(s)=0$$

当 $b_t=0$ 时，一定有 $g(s)=0$。因为 g 的次数比 f 的次数低，这样就能得到一组方程 $g(s)=0$ 并且更易求解。

情形 S3c：存在 g 和 h，h 的次数比 f 的次数低，使得 $f(s)g(s)=h(s)$，这种情形可归结为情形 S3b。假定这种情形成立，则在 $f(s)g(s)=h(s)$ 的两边同乘以 $f(s)$ 可得 $f(s)g(s)=f(s)h(s)=h(s)$，即 $(1\oplus f(s))h(s)=0$，这正是情形 S3b。

情形 S3d：f 存在低次数因子，即存在低次数因子 g 和 h，使得 $f(s)=g(s)h(s)$，这种情形可归结为情形 S3a。假定这种情形成立，则在 $f(s)=g(s)h(s)$ 的两边同乘以 $1\oplus g(s)$ 可得 $f(s)g(s)=g(s)h(s)=f(s)$，即 $(1\oplus g(s))f(s)=0$，因为 $1\oplus g(s)$ 具有低次数，这正是情形 S3a。

显然情形 S3a 和情形 S3b 是相互独立的，二者不相互影响。上述讨论可参阅文献[10]。从上述讨论也可以看出，许多不同的情形可归结为情形 S3a 或情形 S3b，其实从这里也可推出文献[2]中给出的情形 S3 中的 3 种情况可归结为两种情况的结论。这意味着即使 f 的代数次数很高，只要存在一个非零的低次数多项式 g，使得 $fg=0$ 或 $(1\oplus f)g=0$，则代数分析就能有效工作。下面给出一个重要概念——代数免疫度。一个函数 f 的代数免疫度(algebraic immunity，AI)是使得 $fg=0$ 或 $(1\oplus f)g=0$ 成立的非零函数 g 的最小次数，即

$$\mathrm{AI}(f)=\min\{\deg(g)：fg=0 \text{ 或} (1\oplus f)g=0，\quad g\neq 0\}$$

一个函数 f 的代数免疫度刻画了其抵抗代数分析的能力，抵抗代数分析的问题也就变成了寻找具有高的代数免疫度的布尔函数的问题。如果存在 g 使得 $fg=0$，也称 g 为 f 的零化子。

现在的问题是，给定一个密码，这样的多项式 g 是否存在，如何找到它们。这些问题留到后面介绍。

值得一提的是，情形 S1 和 S2 分别是情形 S3 和 S4 的特殊情况，此时 $g=1$。

5. 求解超定多变量方程组

给定 m 个密钥流比特，设 R 是我们获得的次数为 d 的 n 个变量 $k_i(0\leqslant i\leqslant n-1)$ 的多变量方程的个数。对一个 g 而言，可使用 $R=m$，但也许可以用同样的 f 组合一些不同的 g，得到更多的方程，如 $R=14m$。可用如下 3 种方法求解它们。

(1) 线性化方法。有 n 个变量 $k_i(0\leqslant i\leqslant n-1)$、次数不大小 d 的单项式共有 $T\approx C_n^d$ 个(假定 $d\leqslant n/2$)。把这些单项式中的每一个都视作一个新的变量 V_j。给定 $R\geqslant C_n^d$ 个方程，得到一组关于 $T=C_n^d$ 个变量 V_j 的 $R\geqslant T$ 个线性方程，可通过 Gauss 消去法很容易地求解这个有 T 个变量的线性方程组。

(2) XL 算法。$m=O(C_n^d)$ 个密钥流比特是不可能获得的，仍然可能需要使用 XL 算法或 Grobner 基算法求解方程组，这样就需要较少的密钥流，但计算量较大，可参阅文献[5]。

(3) 关于 Gauss 约化的复杂度。设 ω 是 Gauss 约化的指数。理论上，$\omega\leqslant 2.376$[11]。然而在该算法中可忽略的常数因子比预期要大。我们知道的最快的实际算法是 Strassen 算法[12]，需要大约 $7T^{\log_2 7}$ 个操作。因为我们的基本操作是在 F_2 上的，在一个 CPU 上，比特切片技术在单一 CPU 时钟周期内可处理 64 个这样的操作。因此，求得的 Gauss 约化的复杂度是 $7T^{\log_2 7}/64$ 个 CPU 时钟周期。

5.1.2 Toyocrypt 的代数分析

1. Toyocrypt

Toyocrypt 是由一个规则的钟控 128b 线性反馈移位寄存器(LFSR)和一个 127 个变量的非线性布尔函数 f 组成的伪随机序列生成器。函数 f 被用作一个非线性滤波函数，生成器被称为非线性滤波生成器。密钥流通过应用非线性函数到 LFSR 的 128 个段(也称级)中

的 127 个内容来产生。LFSR 每被钟控一次，输出 1b。

Toyocrypt 密钥流生成器有两个 128b 密钥，即一个固定密钥和一个流密钥。固定密钥由 LFSR 的特征多项式(特征多项式与反馈多项式为互反多项式)的系数组成，将这个特征多项式选为 F_2 上的本原多项式，大约有 2^{120} 个这样的多项式。流密钥被用于形成 LFSR 的初始状态，是一个非零的 128b 随机串，大约有 $2^{128}-1$ 个可能的流密钥。因此，有效的密钥长度是 248b。

设 LFSR 的输出序列为 $d=\{d(t)\}_{t=0}^{\infty}$，LFSR 的 128 个段的内容在时刻 t 为 $x_0(t)$，$x_1(t),\cdots,x_{127}(t)$，则 $d(t)=x_0(t-1),t\geqslant 1$。将输入到滤波函数 f 的序列表示为 $X=\{X(t)\}_{t=0}^{\infty}$，这里 $X(t)=(x_0(t),x_1(t),\cdots,x_{125}(t),x_{127}(t))$。注意，$x_{126}(t)$没有作为 f 的输入。非线性滤波生成器的输出序列是密钥流 $z=\{z(t)\}_{t=0}^{\infty}$，这里 $z(t)=f(X(t)),t\geqslant 0$。

布尔函数 f 的代数正规型表示如下：

$$\begin{aligned}f(x_0,x_1,\cdots,x_{125},x_{127})=&x_{127}+x_{24}x_{63}+x_{41}x_{64}+x_{27}x_{65}+x_{32}x_{66}+x_{35}x_{67}+x_{50}x_{68}+\\&x_8x_{69}+x_{18}x_{70}+x_1x_{71}+x_{36}x_{72}+x_{53}x_{73}+x_{26}x_{74}+x_3x_{75}+\\&x_7x_{76}+x_{11}x_{77}+x_6x_{78}+x_{62}x_{79}+x_{37}x_{80}+\\&x_{31}x_{81}+x_{12}x_{82}+x_9x_{83}+x_{34}x_{84}+x_{51}x_{85}+x_{61}x_{86}+\\&x_{25}x_{87}+x_{23}x_{88}+x_{45}x_{89}+x_{14}x_{90}+x_0x_{91}+x_{20}x_{92}+\\&x_{46}x_{93}+x_{38}x_{94}+x_{40}x_{95}+x_{13}x_{96}+x_{28}x_{97}+x_2x_{98}+\\&x_{49}x_{99}+x_{54}x_{100}+x_5x_{101}+x_{60}x_{102}+x_{47}x_{103}+x_4x_{104}+\\&x_{16}x_{105}+x_{52}x_{106}+x_{59}x_{107}+x_{55}x_{108}+x_{10}x_{109}+x_{57}x_{110}+\\&x_{22}x_{111}+x_{15}x_{112}+x_{56}x_{113}+x_{48}x_{114}+x_{29}x_{115}+x_{44}x_{116}+\\&x_{39}x_{117}+x_{58}x_{118}+x_{17}x_{119}+x_{43}x_{120}+x_{19}x_{121}+\\&x_{21}x_{122}+x_{42}x_{123}+x_{33}x_{124}+x_{30}x_{125}+\\&x_{10}x_{23}x_{32}x_{42}+x_1x_2x_9x_{12}x_{18}x_{20}\ x_{23}x_{25}x_{26}\\&x_{28}\ x_{33}x_{38}x_{41}x_{42}x_{51}x_{53}x_{59}+x_0x_1\cdots x_{62}\end{aligned}$$

2. Toyocrypt 的代数分析过程

在 Toyocrypt 中，有一个 128b 的 LFSR，这样 $n=128$。布尔函数具有如下形式：

$$f(s_0,s_1,\cdots,s_{127})=s_{127}+\sum_{i=0}^{62}s_is_{\alpha_i}+s_{10}s_{23}s_{32}s_{42}+$$

$$s_1s_2s_9s_{12}s_{18}s_{20}s_{23}s_{25}s_{26}s_{28}s_{33}s_{38}s_{41}s_{42}s_{51}s_{53}s_{59}+\prod_{i=0}^{62}s_i$$

这里$\{\alpha_0,\alpha_1,\cdots\alpha_{62}\}$是集合$\{62,64,\cdots,125\}$的某一置换。这个系统容易受使用低阶逼近的攻击，只有一个次数为 17 的单项式和一个次数为 63 的单项式，高阶单项式几乎总是 0。文献[5]在情形 S2 下给出了一个攻击，f 由一个次数为 4 的多变量函数以 $1-2^{-17}$ 的概率逼近，攻击运行在 2^{92} 个 CPU 时钟周期内。下面在情形 S3 或 S4 下介绍对 Toyocrypt 的一个新的代数分析方法。

首先遇到的问题是如何找到一个函数 g 使得 fg 具有低次数。一种方法是分解多变量多项式 f。考虑 f 的高次项，不管低次项，看看这些高次项是否有共同的低次数因子 g'，则对 F_2 上的多项式，我们观察到 $f(s)g(s)$，其中 $g(s)=g'(s)-1$ 具有低次数。另一种方法

将在 5.1.3 节和 5.1.4 节介绍。

在 Toyocrypt 的情况下，我们观察到 f 中的次数为 4、17 和 63 的项有共同因子 $s_{23}s_{42}$。对每个密钥流比特 b_t，从等式 $f(s)=b_t$ 开始，并在两边同乘以 $g(s)=s_{23}-1$。得到 $f(s)s_{23}-f(s)=b_t(s_{23}-1)$。等式左边在 f 中被 s_{23} 整除的单项式已消失，剩下的多项式是一个次数为 3 的等式且以概率 1 相等。对 s_{42} 使用相同的技巧，即设 $g(s)=s_{42}-1$。这样就得到情形 S3 下的一个简单的线性攻击。对每个密钥流比特，获得两个次数为 3 的关于 s_i 的方程，这样也就获得了两个次数为 3 的关于 k_i 的方程。一旦 $R>T$，就可使用线性方法，并且有 $T=C_{128}^3\approx 2^{18.4}$ 个单项式。有 $R=2m$，并且有 $m=T/2\approx 2^{17.4}$ 个密钥流将是充分的。这样，对 Toyocrypt 的新的攻击需要 $7/64\cdot T^{\log_2 7}=2^{49}$ 个 CPU 时钟周期，需要 16GB 的存储空间和大约 20KB 的非连续的密钥流。

通过实验模拟可发现，上述攻击过程中的线性依赖的方程的个数是可忽略的，因此，这个攻击可恢复密钥。

5.1.3 LILI-128 的代数分析

从原理上说，前面设计的代数分析只适用于规则地钟控的序列密码或以一个已知的方式钟控的序列密码。然而，在一些情况下，这个难点能被消除。LILI-128 就是这种情况，LILI-128 是一个提交到欧洲 NESSIE 工程的算法。

1. LILI-128

LILI-128 密钥流生成器的结构如图 5.1.1 所示。它由两个二元 $LFSR_c$ 和 $LFSR_d$ 以及生成一个伪随机二元密钥流序列的两个函数 f_c 和 f_d 组成。在初始化阶段，128 比特密钥为 $LFSR_c$ 和 $LFSR_d$ 提供了初始状态。基于它们完成的功能，生成器能被分成两个子系统：钟控子系统和数据生成子系统。钟控子系统产生一个用于控制数据生成子系统的整数序列。$LFSR_c$ 的反馈多项式被选择为一个本原多项式，定义为

$$G_c(x)=1+x^2+x^{14}+x^{15}+x^{17}+x^{31}+x^{33}+x^{35}+x^{39}$$

函数 f_c 取两个比特作为输入并产生一个整数 $c_k\in\{1,2,3,4\}$。c_k 的值计算如下：

$$c_k=f_c(y_1,y_2)=2y_1+y_2+1,\quad k\geqslant 1$$

这里变量 y_1、y_2 分别对应于 $LFSR_c$ 的抽头位置 s_{i+13}、$s_{i+21}(i\geqslant 0)$，需要注意的是 f_c 中的＋是普通加法，其他＋是模 2 加。

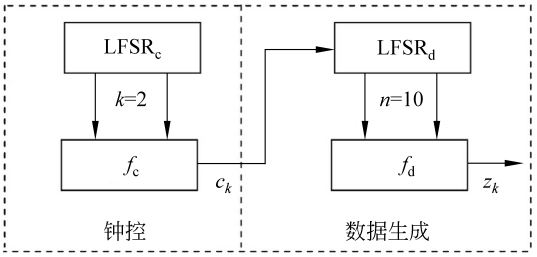

图 5.1.1 LILI-128 密钥流生成器的结构

$LFSR_d$ 由 c_k 钟控每次运动至少一拍，至多 4 拍。$LFSR_d$ 的反馈多项式被选择为如下的本原多项式：

$$G_d(x)=1+x+x^{39}+x^{42}+x^{53}+x^{55}+x^{80}+x^{83}+x^{89}$$

$LFSR_d$ 的 10 个不同段的内容被作为一个非线性滤波函数 f_d 的输入。非线性滤波函数 f_d 的输出 z_k 是密钥流序列。f_d 定义为

$$\begin{aligned}f_d=&x_2+x_3+x_4+x_5+x_1x_9+x_1x_8+x_2x_8+x_3x_9+x_4x_{10}+x_6x_7+x_6x_{10}+\\&x_2x_9x_{10}+x_3x_9x_{10}+x_4x_9x_{10}+x_5x_9x_{10}+x_3x_8x_{10}+x_4x_8x_{10}+x_6x_7x_{10}+\\&x_5x_7x_{10}+x_4x_7x_{10}+x_6x_8x_9+x_3x_8x_9+x_6x_7x_9+x_4x_7x_9+x_3x_7x_9+\\&x_6x_8x_9x_{10}+x_4x_8x_9x_{10}+x_3x_8x_9x_{10}+x_1x_8x_9x_{10}+x_6x_7x_9x_{10}+x_4x_7x_9x_{10}+\\&x_2x_7x_9x_{10}+x_5x_7x_8x_{10}+x_3x_7x_8x_{10}+x_4x_7x_8x_9+x_2x_7x_8x_9+x_5x_6x_7x_9+\\&x_4x_6x_7x_9+x_3x_7x_8x_9x_{10}+x_4x_6x_7x_8x_9+x_4x_6x_7x_9x_{10}+x_4x_7x_8x_9x_{10}+\\&x_5x_6x_7x_8x_9+x_5x_6x_7x_9x_{10}+x_4x_6x_7x_8x_9x_{10}+x_5x_6x_7x_8x_9x_{10}\end{aligned}$$

这里变量 $x_1\sim x_{10}$ 分别对应于 $LFSR_d$ 的抽头位置 s_i、s_{i+1}、s_{i+3}、s_{i+7}、s_{i+12}、s_{i+20}、s_{i+30}、s_{i+44}、s_{i+65}、s_{i+80}($i\geqslant 0$)。

设 $LFSR_c$ 的段由 $u_0,u_1,\cdots,u_{37}$ 从左到右标记。在每个时刻,用下列公式计算反馈比特:

$$w=u_{37}+u_{25}+u_{24}+u_{22}+u_8+u_6+u_4+u_0$$

设 $LFSR_c$ 左移,$u_{38}=w$。连续地循环,$LFSR_c$ 将产生一个线性伪随机序列。

设 $LFSR_d$ 的段由 $s_0,s_1,\cdots,s_{88}$ 从左到右标记。在时刻 k,反馈比特由下列公式计算:

$$w=s_{88}+s_{50}+s_{47}+s_{36}+s_{34}+s_9+s_6+s_0$$

设 $LFSR_d$ 左移,$u_{89}=w$。根据上述循环,$LFSR_d$ 将产生一个线性伪随机序列。

2. 消除第一个组件

LILI-128 是由两个基于 LFSR 的组件组成的序列密码,第一个组件被用于钟控第二个组件。有两种策略可绕过第一个组件。

策略 A:因为第一个组件的密钥长度仅是 39b,可以猜测这 39b 并单独地攻击第二个组件。在 LILI-128 中,第一个组件提供给第二个组件的时钟为 1、2、3 或 4。给定第一个组件的状态,访问第二个组件中若干已知位置的非连续密钥流比特。攻击的复杂度是 2^{39} 的倍数。

策略 B:给定更多的密钥流比特,有可能避免重复整个攻击 2^{39} 次。对此,使用文献[13]中的引理 1:在对 LILI-128 的第一个 LFSR 钟控 $2^{39}-1$ 次后,第二个 LFSR 恰好向前推进 $\Delta_d=5\cdot 2^{38}-1$ 次。这样,就不是猜测时钟控制的子系统的状态,而是在一个时刻钟控它 $2^{39}-1$ 次,并应用任何 XL 攻击,忽略第一个生成器的存在。

在上述两种情况下,攻击者访问的是第二个组件的密钥流中一些已知位置的比特,而不是自己选择位置的比特。可直接将第二个组件当作一个单独的滤波生成器,应用情形 S1 至情形 S4 的所有代数分析。

中间策略 A-B:LILI-128 的第一个组件的周期不是一个素数,并且有 $2^{39}-1=7\cdot 79\cdot 121\ 369\cdot 8191$。这暗示着人们能设计一个抽取攻击,对于一个适当的 t,令生成器每次运行 t 拍,可模拟一个较小的 LFSR,详细讨论可参阅文献[14]。可给出一个在 A 和 B 之间的中间攻击,密钥流需求和攻击复杂度都是某个量的倍数,都小于 2^{39}。

上面和文献[15]中都将 LILI-128 中的输出函数记为 f_d,为了方便起见,下面将其记

为 f。

3. 对 LILI-128 的第一个攻击

现在讨论在情形 S1 和 S2 下对 LILI-128 的攻击。在情形 S1 下，攻击是可能的，此时 $d=6,\varepsilon=0,R=m$，则有 $T=C_{89}^{6}\approx 2^{29.2}$ 个单项式。为了使得 $R>T$，取 $m\approx 2^{29.2}$。可给出一个大约在 $2^{39}\cdot 7/64\cdot T^{\log_2 7}\approx 2^{118}<2^{128}$ 个 CPU 时钟周期内的攻击。给定使用的布尔函数，不适合将这个攻击扩展到情形 S2 下，因为对给定的 $\varepsilon>2^{-6}$，没有次数小于 6 的好的逼近，低次数的多项式与要逼近的函数的距离比较大。然而，可以用策略 B 改进这个攻击。改进的攻击不是猜测钟控子系统的状态，而是在一个时刻钟控它 $2^{39}-1$ 次，并应用上述简单的线性化 S1 型攻击，$\varepsilon=0$，而可忽略第一个生成器的存在。现在的复杂度仅是 $7/16\cdot T^{\log_2 7}\approx 2^{79}$，但需要更多的密钥流比特，即 2^{68} b 而不是 2^{29} b。

4. 对 LILI-128 较好的攻击

先使用前面提到的分解多变量多项式的观点对 LILI-128 进行分析。考虑 f 的次数为 5 和 6 的部分，它可被分解成如下形式：

$$x_7x_9(x_3x_8x_{10}+x_4x_6x_8+x_4x_6x_{10}+x_4x_8x_{10}+x_5x_6x_8+x_5x_6x_{10}+x_4x_6x_8x_{10}+x_5x_6x_8x_{10})$$

这意味着当 f 乘以 x_7+1 或 x_9+1 时，得到的多项式的次数从 $6+1=7$ 降到 $4+1=5$。然后，分别考虑多项式 $f(x)(x_7+1)$ 和 $f(x)(x_9+1)$ 中次数为 5 和 4 的部分的分解。只有第二个函数的相应部分仍然能被分解并可分解为如下形式：

$$x_{10}(x_3x_7x_8x_9+x_5x_7x_8x_9+x_3x_7x_8+x_3x_8x_9+x_4x_7x_9+x_4x_8x_9+x_5x_7x_8+x_5x_7x_9+x_6x_7x_9)$$

由此可推断出显著的事实：$f(x)(x_9+1)(x_{10}+1)$ 的次数是 4 而不是 8。

为了找到使得 fg 具有低次数的线性独立的 g 的个数，我们寻找下列多项式集合的线性依赖性（在 g 的最大次数和 fg 的某一最大次数终止）：

$$\{f(x),f(x)x_1,f(x)x_2,\cdots,f(x)x_1x_2,\cdots;1,x_1,x_2,\cdots,x_1x_2,\cdots\}$$

我们注意到最大次数不可能高于 10，因为仅有 10 个变量。表 5.1.1 给出了 $fg\neq 0$ 的相关模拟结果，并与 g 的次数相同的随机布尔函数进行了比较。通过计算和测试可知

$$f(x)x_8x_{10}=x_8x_{10}(x_2x_9+x_3x_7+x_4x_7+x_5x_9+x_1+x_4+x_5+x_6)$$

表 5.1.1 使得 fg 具有低次数的线性独立的 g 的个数的模拟结果

函 数	LILI-128 的 f						随机布尔函数					
g 的次数	10	1	2	3	4	10	10	1	2	3	4	10
fg 的次数	3	4	4	4	4	4	3	4	4	4	4	4
g 的个数	0	0	4	8	14	14	0	0	0	0	0	0

我们看到，LILI-128 的函数 f 与随机布尔函数相比性质很弱。这表明 LILI-128 的设计不足以抵抗代数分析。事实上，在 LILI-128 中，也存在次数为 4 的多项式使得 $fg=0$。

给定 m 个密钥流比特，可获得 LILI-128 的次数为 4 的 $14m$ 个关于密钥流比特 k_i 的多变量方程。我们将不得不解决一个次数为 4 的超定多变量方程组，这个方程组是真的概率

为1,可由线性化方法来完成。对应于上述两种策略(即策略A和B)有两种攻击形式。

在攻击形式A中,第一个生成器的状态被猜测并且复杂度是2^{39}的倍数。对每个密钥流比特,我们获得14个次数为4的关于k_j的方程。为了线性化,有$T=C_{89}^{4}\approx 2^{21}$个单项式,并且为了使$R>T$,需要$m=T/14\approx 2^{18}$个密钥流比特。对LILI-128的这个攻击需要$2^{39}\cdot 7\cdot T^{\log_2 7}/64\approx 2^{96}$个CPU时钟周期。只要允许访问在一些已知位置的$m$个密钥流比特,这一攻击形式就有效。

在攻击形式B中,在某个时刻第一部分被钟控$2^{39}-1$个时钟周期,密钥流比特的数量是2^{39}的倍数。有同样的$T=C_{89}^{4}\approx 2^{21}$,复杂度是$2^{57}$个时钟周期。这种攻击需要762GB的存储空间和$2^{57}$个连续的密钥流比特,或者只需要在形式为$\alpha+\beta(2^{39}-1)$(对一个固定的$\alpha$)的一些位置的$2^{18}$个密钥流比特。一旦第二个LFSR在时刻$\alpha$的状态被恢复,则第一个LFSR的状态在$2^{39}$次尝试内能被很快地找到。这不是对LILI-128的最好的已知攻击,文献[13]中的分析结果表明,LILI-128可用2^{46}个密钥流比特、2^{45}个89b字的查表和大约等效于2^{48}个DES操作的计算努力破译。然而,攻击形式B更具有一般性。

5.1.4 使用LFSR比特的一个子集对序列密码的一般攻击

本节将说明当只使用状态比特的一个小子集时,由线性反馈和一个无状态的高度非线性滤波函数组成的生成器是不安全的。考虑一个具有n个状态比特的序列密码,并且只使用k个状态比特的小子集来推导密钥流比特。这样,就有$\{x_1,x_2,\cdots,x_k\}\subset\{s_0,s_1,\cdots,s_{n-1}\}$,这里假定$k$是一个小常数,$n$是一个安全参数。例如,对LILI-128的第二个部分,$k=10$,$n=89$。

这里将在情形S3下进行一个攻击。寻找低次数多项式$g\neq 0$使得fg也具有低次数。为了达到这一目标,检查定义在下列的多项式集$C=A\cup B$上的线性依赖性。首先考虑次数不大于d的所有可能的单项式(这一部分将构成fg),记为$A=\{1,x_1,x_2,\cdots,x_1x_2,\cdots\}$。然后考虑次数不大于$d$的$f$的所有乘以单项式的倍式(这一次数对应$g$的次数),记为$B=\{f(x),f(x)x_1,f(x)x_2,\cdots,f(x)x_1x_2,\cdots\}$。

设$C=A\cup B$。A、B、C中的所有元素可被视作x_i的多变量多项式,因此需要用x_i的表达式代替f。具有k个变量的多变量多项式集合的维数不能大于2^k。如果在这个集合中的元素多于2^k个,则线性依赖性必将存在。这种组合允许找到一个函数g使得fg具有比f低的次数。更精确地,有如下的低次数关系定理。

定理5.1.1 设$f:F_2^k\to F_2$是任意一个布尔函数,则有一个次数至多为$\lceil k/2\rceil$的布尔函数$g\neq 0$,使得fg的次数至多为$\lceil k/2\rceil$。

证明:如果A中包括所有次数直到$\lceil k/2\rceil$的单项式,则$|A|=\sum_{i=0}^{\lceil k/2\rceil}C_k^i$。类似地,如果$B$中包括所有次数直到$\lceil k/2\rceil$的单项式与$f$的积,则$|B|=\sum_{i=0}^{\lceil k/2\rceil}C_k^i$。所以$|C|=|A|+|B|=\sum_{i=0}^{\lceil k/2\rceil}C_k^i+\sum_{i=0}^{\lceil k/2\rceil}C_k^i=\sum_{i=0}^{\lceil k/2\rceil}C_k^i+\sum_{i=0}^{\lceil k/2\rceil}C_k^{k-i}>\sum_{i=0}^{k}C_k^i=2^k$,而$C=A\cup B$的个数不能超过$2^k$,这样就必然存在一定的线性依赖性。因为在集$C$中的$A$部分没有线性依赖性,所以$g\neq 0$。

由定理5.1.1可知,对于任何基于线性反馈的非线性滤波序列密码,如果非线性滤波使

用 k 个变量，则在 n 个密钥流比特中，可以至少产生一个次数约为 $k/2$ 的方程。这些方程通常可由线性化方法来解决。为了获得一个完全饱和的可通过线性化方法求解的方程组，需要至多 $\sum_{i=0}^{k/2} C_n^i \approx C_n^{k/2}$ 个密钥流比特。再者，如果对一个给定的 f，在 C 中有一定的线性依赖性，则可以使用一些线性独立的 g，并对于每个密钥流比特将获得相应个数的方程。因而，密钥流长度要求必须被整除，例如前面介绍的例子可被 14 整除。

综上所述，可得到如下的对任意 k 个输入的布尔函数的一般化攻击：$d=k/2$，$\varepsilon=0$，数据量为 $C_n^{k/2}$，存储量为 $(C_n^{k/2})^2$，复杂度为 $(C_n^{k/2})^\omega$。这个攻击可处理最坏的情况。而具体密码使用的函数有可能具有更低的安全性。例如，在 LILI-128 中，$k=10$，严格应用定理 5.1.1，最坏情况下的复杂度是 $O(n^{6\omega})$（对任意的布尔函数）。而在文献[2]的扩展版本中证明其平均复杂度是 $O(n^{5\omega})$（对一个随机布尔函数）。对于使用在 LILI-128 中的具体函数，前面介绍的攻击的复杂度是 $O(n^{4\omega})$。

如果 k 是固定的，那么这个攻击是多项式的。如果 $k=O(n)$，这个攻击关于 n 仅是指数的。使用在一个滤波函数中的比特数不会小。在实际中，谈论多项式或非多项式时间容易引起误解并且总是用具体的结果来控制。假定知道滤波函数的最大次数不能超过 k，则可看到在情形 S1 下，基于线性反馈的任何序列密码都能通过简单的线性化方法被破译，需要给定 C_n^k 个密钥流比特，复杂度大约为 $(C_n^k)^\omega$。如果 k 是固定的，这个简单的攻击方法是多项式的[15]。许多以这种方式设计的序列密码的复杂度大约为 $(C_n^k)^\omega \approx 2^{80}$。实际中，攻击的复杂度 $(C_n^{k/2})^\omega$ 大约是 $(C_n^k)^\omega \approx 2^{80}$ 的平方根，所以破译所有这些密码的复杂度大约是 2^{40}。

5.1.5 应对代数分析方法的措施

由定理 5.1.1 可得出如下实现最优抵抗代数分析的要求：当集合 A 和 B 被生成直到任何严格地小于依赖性一定存在的次数时，确保没有线性依赖性存在。这可归结为如下的准则。

最优抵抗准则 5.1.1 当集合 A 和 B 被生成直到关于 x_i 的次数为 $\lfloor k/2 \rfloor$，则集合 $C=A \cup B$ 的个数是最大的。

易知，这个准则暗含着要阻止情形 S1 的攻击形式，f 的次数将是充分大的。然而，这个要求不能保证情形 S2 或 S4 的攻击不存在[5]。

综合前面的分析结果可以看出，具有下列情况之一的滤波生成器都可用代数分析方法来攻击：

(1) f 使用一个小的状态比特子集，例如 LILI-128 中使用的是 10b 的子集。

(2) f 是很稀疏的，如 Toyocrypt 中使用的函数。

(3) 能从 f 中分解出一个低次数因子，如 Toyocrypt 中使用的函数。

(4) 能用上述情况之一来逼近 f。

(5) f 的高次数部分是上述情况之一。

从上述情况可以得出以下结论。首先，在滤波生成器中，滤波函数应使用许多状态比特（如至少 32b）并且不是太稀疏的，所以它也有许多很高次数的项（如至少 32 项）。其次，高次数部分不具有低次数因子，并且它本身也使用了许多状态比特。最后，高次数部分的近似逼近同样不能具有低次数因子，或者只使用了一小部分状态比特。从 LILI-128 中可以看