

减轮 Fruit 算法的 Cube 攻击

孙移盛

清华大学 计算机科学与技术系, 北京 100084

通讯作者: 孙移盛, E-mail: sun-ys14@mails.tsinghua.edu.cn

摘要: Cube 攻击是由 Dinur 和 Shamir 在 2009 年的欧密会上提出的一种代数攻击方法, 它旨在从目标加密算法中提取关于未知变量的线性关系, 而难点是在寻找有效的 Cube. 超轻量级序列密码具有速度快, 功耗低, 便于硬件实现等优点, 市场对超轻量级序列密码的需求很大, 使得密码学界对超轻量级序列密码算法的研究更加深入. 2015 年 Armknetcht 等人对轻量级序列密码提出了一个新的设计方向, 在每轮密钥流比特的生成过程中重复使用初始密钥比特, 并基于此想法设计了一个超轻量级序列密码算法 Sprout, 使得内部状态大小与密钥大小相同, 打破了超轻量级序列密码设计的瓶颈. Fruit 是 2016 年由 Ghafari 等设计的一种超轻量级流密码, 其设计目的是在保证内部状态很小的同时能避免时间-存储-数据折衷攻击. 本文对减轮的流密码 Fruit 作 Cube 攻击, 在随机选取 Cube 方法的基础上提出一些寻找 Cube 的新想法, 并最终对减轮的 83 轮 (最高可到 86 轮) Fruit 算法做 Cube 攻击求得 80 个密钥中的 17 个密钥, 比穷尽搜索降低了 2^{17} 的复杂度. 并发现找到的线性多项式只与密钥的后 17 比特有关, 没有发现关于密钥前 63 比特的线性表示, Fruit 算法的轮密钥函数导致的结果, 对轮密钥函数的分析有很好的借鉴意义.

关键词: 流密码; Fruit 算法; Cube 攻击

中图分类号: TP309.7 文献标识码: A DOI: 10.13868/j.cnki.jcr.000204

中文引用格式: 孙移盛. 减轮 Fruit 算法的 Cube 攻击[J]. 密码学报, 2017, 4(6): 528–536.

英文引用格式: SUN Y S. Cube attack on round-reduced Fruit[J]. *Journal of Cryptologic Research*, 2017, 4(6): 528–536.

Cube Attack on Round-reduced Fruit

SUN Yi-Sheng

Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

Corresponding author: SUN Yi-Sheng, E-mail: sun-ys14@mails.tsinghua.edu.cn

Abstract: The cube attack is an algebraic cryptanalysis method introduced by Dinur and Shamir at EUROCRYPT 2009, it aims to extract linear relations about secret variables from the targeted primitives, and the difficulty is to find a good cube. Ultra-lightweight stream ciphers have the advantages of high speed, low power consumption, easy to implement and so on. The market demand for ultra-lightweight stream ciphers is very large, which makes the ultra-lightweight stream ciphers to be more attractive. In 2015, Armknetcht et al. proposed a new design direction for lightweight stream ciphers, with repeated use of initial key bits in each round of key stream bit generation. Based on this idea, they proposed a new ultra-lightweight stream cipher named Sprout, with the internal state size and the key size are both 80 bits. An ultra-lightweight stream cipher Fruit was designed by Ghafari et al. in 2016 to reduce the internal state without harming its security against time-memory-data tradeoff

attack. This work applies the cube attack to round-reduced Fruit. We consider a few new ideas for obtaining good cubes. The cube attack on round-reduced Fruit (83 rounds, up to 86 rounds) can recover 17 bits of keys out of the 80-bit key, which is 2^{17} times faster than exhaustive key search. The linear polynomial is found to be only related to the last 17 bits of the key, no linear representation is found in the first 63 bits of the key. This result is due to the round key function of the Fruit algorithm, it is a good reference for the analysis of the round key function.

Key words: stream cipher; Fruit; Cube attack

1 引言

Cube 攻击是基于代数结构的选择明文攻击, 由 Dinur 和 Shamir 在 2009 年的欧密会上提出^[1]. 该方法继承了高阶差分攻击^[2]和 IV 代数差分攻击^[3]思想, 其攻击原理是将密码算法描述为有限域 $GF(2)$ 上关于未知变量 k (如密钥) 和公开变量 v (如 IV, 明文) 的多项式 $P(v, k)$, 通过选取 P 的部分变量组成 Cube, 将 Cube 的所有可能值输入 $P(v, k)$ 得到关于 k 的低次方程, 由不同的 Cube 得到关于 k 的低次方程组, 通过求解方程组恢复 k . Cube 攻击适用很多密码算法, 对分组密码和序列密码都有攻击效果, 一经提出就得到了广泛的应用, 最初应用于流密码 Trivium^[4,5], 随后又分析了 Grain-128^[6], CTC^[7] 等. 本文分析 Cube 攻击方法攻击流密码 Fruit^[8] 算法的可能性.

Fruit^[8] 算法是由 Vahid Amin Ghafari 等在 2016 年提出的一个超轻量级流密码. 在现代密码学中, 轻量级流密码起着至关重要的作用, 最近几年在处理能力较低的设备中得到广泛应用如 WSN, RFID 等, 具有低功耗, 占用硬件面积小等优点, 因此市场对轻量级密码的需求很大, 在过去十年中提出了许多新的密码算法, 如 Grain^[9], Trivium^[4], LBlock^[10], PRESENT^[11], Klein^[12] 等.

1999 年 NESSIE 项目的失败导致欧盟在 2004 年推出 eSTREAM^[13], 该项目包括硬件和软件两个方面. 在 eSTREAM 的第一阶段提出了总共 34 个密码算法, 但大部分都被淘汰, 只有少数密码算法延续到了下一阶段. 在最后一次修订后, 硬件类别中只有 Mickey^[14], Grain-v1^[9] 和 Trivium^[4] 仍然保留. 然而这些密码算法的内部状态还是比较大, 轻量级流密码的设计需要更小的内部状态, 但为了避免时间-存储-数据折衷 (TMD) 攻击^[15], 流密码的通常设计原则是保持内部状态大小至少是密钥大小的两倍, 这对轻量级流密码的设计提出了更高的挑战. 2015 年, Armknetcht^[16] 等人针对轻量级流密码提出了新的设计思路, 希望既减小了内部状态又能避免 TMD 攻击. Armknetcht 等人建议密钥不仅在初始化过程中使用, 在加密时也重复使用它, 基于此, 他们提出了一个新的密码算法 Sprout^[16], 使用 80 比特密钥, 而其内部状态大小也是 80 比特, 但令人遗憾的是 Sprout 不安全^[17-19]. 尽管如此, 但是该想法还是打破了轻量级流密码的设计瓶颈, 基于相同的想法, 2016 年 Hamann 等提出了 Lizard^[20], 它使用 120 位的密钥和 121 位的内部状态, Mikhalev 等提出了 Plantlet^[21] 算法, 它使用 80 位密钥, LFSR 和 NFSR 大小分别为 61 和 40. Fruit 的设计思想也是基于这一想法, 并且针对 Sprout 的攻击设计了更复杂的轮密钥函数, 使用更大的 LFSR 并避免初始化后 LFSR 状态全 0. 根据作者所说, Fruit 算法在抗 Cube 攻击, TMD 攻击方面比 Grain 和 Sprout 更安全.

本文的主要贡献是首次用 Cube 攻击对 Fruit 算法作安全性分析, 在随机选取 Cube 方法的基础上提出增减有效 Cube 的变量, 重新组合有效 Cube 等寻找 Cube 的新想法, 并发现 Cube 指标及其对应的轮数和密钥的线性表达式有很好代数递推关系, 提高寻找 Cube 的效率. 用 Cube 攻击方法对减轮 Fruit 算法攻击恢复了 17 比特密钥, 比穷尽搜索整个密钥空间降低 2^{17} 的复杂度.

本文结构安排如下: 第2节讨论 Fruit 的结构, 它是一个类 Grain 的设计结构; 第3节介绍 Cube 攻击原理, 给出了 Cube 攻击的具体步骤; 第4节给出了 Cube 攻击分析 Fruit 的细节; 最后第5节对全文进行总结.

2 Fruit 介绍

这一节对 Fruit 算法的设计做简要介绍. Fruit 是超轻量级流密码 (如图1), 该密码设计者的设计目的是在保证内部状态很小的同时能避免 TMD 攻击, 其内部状态是 80 比特, 由 43 比特线性移位寄存器

(LFSR) 和 37 比特非线性移位寄存器 (NFSR) 组成, 使用 80 比特密钥 K 和 70 比特初始化向量 IV 作为输入. 注意到在单个 K 和 IV 的情况下最多可以产生 2^{43} 比特密钥流, 因此为了安全起见, 算法设计者建议对某个 K 的使用次数应少于 2^{15} 并且 IV 不能重用.

为更好地理解 Fruit 算法, 对一些符号做如下说明:

t : 轮数

L_t : 第 t 轮的 LFSR 状态 $(l_t, l_{(t+1)}, \dots, l_{(t+42)})$

N_t : 第 t 轮的 NFSR 状态 $(n_t, n_{(t+1)}, \dots, n_{(t+36)})$

C_r : 7 比特计数器 $(c_t^0, c_t^1, \dots, c_t^6)$

C_c : 7 比特计数器 $(c_t^7, c_t^8, \dots, c_t^{14})$

k : 密钥 $(k_0, k_1, \dots, k_{79})$

k'_t : 第 t 轮由密钥函数产生的密钥

IV : 初始化向量 $(v_0, v_1, \dots, v_{69})$

z_t : 第 t 轮生成的密钥流比特

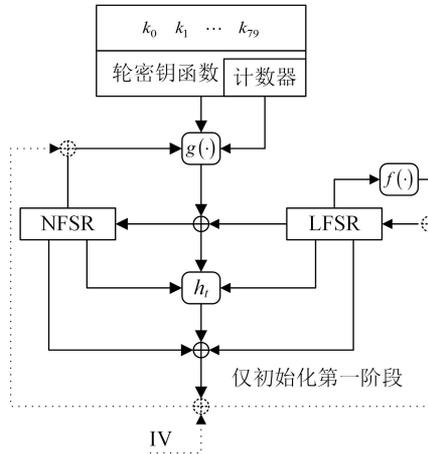


图 1 流密码 Fruit 算法

Figure 1 Structure of stream cipher Fruit

计数器: Fruit 算法的 15 比特计数器由独立的两部分组成, 7 比特的 C_r 和 8 比特的 C_c , 分别用于生成轮密钥和密钥流, 均从 0 开始计数, 每轮增 1.

轮密钥函数: 轮密钥由 6 比特密钥生成, 其生成函数为

$$k'_t = k_s k_{(y+64)} \oplus k_{(u+72)} k_p \oplus k_{(q+32)} \oplus k_{(r+64)} \quad (1)$$

其中 $s = (c_0^t c_1^t c_2^t c_3^t c_4^t c_5^t)$, $y = (c_3^t c_4^t c_5^t)$, $u = (c_4^t c_5^t c_6^t)$, $p = (c_0^t c_1^t c_2^t c_3^t c_4^t)$, $q = (c_1^t c_2^t c_3^t c_4^t c_5^t)$, $r = (c_3^t c_4^t c_5^t c_6^t)$.

LFSR: 43 比特线性移位寄存器反馈多项式为

$$l_{(t+43)} = f(L_t) = l_t \oplus l_{(t+8)} \oplus l_{(t+18)} \oplus l_{(t+23)} \oplus l_{(t+28)} \oplus l_{(t+37)} \quad (2)$$

NFSR: 37 比特非线性移位寄存器反馈输入包括计数器的 1 比特, k'_t , l_t 及 NFSR 的 16 比特, 其多项式表示为

$$\begin{aligned} n_{(t+37)} &= k'_t \oplus l_t \oplus c_t^{10} \oplus g(N_t) \\ &= k'_t \oplus l_t \oplus c_t^{10} \oplus n_t \oplus n_{(t+10)} \oplus n_{(t+20)} \oplus n_{(t+12)} n_{(t+3)} \end{aligned}$$

$$\begin{aligned} &\oplus n_{(t+14)}n_{(t+25)} \oplus n_{(t+5)}n_{(t+23)}n_{(t+31)} \oplus n_{(t+8)}n_{(t+18)} \\ &\oplus n_{(t+28)}n_{(t+30)}n_{(t+32)}n_{(t+34)} \end{aligned} \tag{3}$$

输出函数: 输出函数的输出结果 z_t 由 LFSR 的 1 比特, NFSR 的 7 比特与非线性函数 h_t 异或得到, 其计算公式为

$$\begin{aligned} z_t &= n_t \oplus n_{(t+7)} \oplus n_{(t+13)} \oplus n_{(t+19)} \oplus n_{(t+24)} \oplus n_{(t+29)} \oplus n_{(t+36)} \oplus n_{(t+38)} \oplus h_t \\ &= n_t \oplus n_{(t+7)} \oplus n_{(t+13)} \oplus n_{(t+19)} \oplus n_{(t+24)} \oplus n_{(t+29)} \oplus n_{(t+36)} \\ &\quad \oplus n_{(t+38)} \oplus n_{(t+1)}l_{(t+15)} \oplus l_{(t+1)}l_{(t+22)} \oplus n_{(t+35)}l_{(t+27)} \\ &\quad \oplus n_{(t+33)}l_{(t+11)} \oplus l_{(t+6)}l_{(t+33)}l_{(t+42)} \end{aligned} \tag{4}$$

密码初始化: 将 IV 扩展为 130 比特 IV', 即在 IV 的前面添加 1 比特 1 和 9 比特 0, 在 IV 的后面添加 50 比特 0. 形如

$$IV' = 1000000000v_0v_1 \cdots v_{69}00 \cdots 00$$

在初始化阶段, 80 比特密钥 K 载入 LFSR 和 NFSR, K 的前 37 比特载入 NFSR 即 $n_0 = k_0, n_1 = k_1, \cdots, n_{36} = k_{36}$, 剩下的 43 比特载入 LFSR 即 $l_0 = k_{37}, l_1 = k_{38}, \cdots, l_{42} = k_{79}$, 两个计数器均置 0, 初始化为两个阶段.

初始化第一阶段要将 z_t 异或 IV' 的比特参与 LFSR 和 NFSR 的反馈:

$$\begin{aligned} l_{(t+43)} &= z_t \oplus v'_t \oplus f(L_t) \\ n_{(t+37)} &= z_t \oplus v'_t \oplus k'_t \oplus l_t \oplus c_t^{10} \oplus g(N_t) \end{aligned}$$

该阶段要运行 130 轮转入第二阶段, 每跑一轮计数器 C_r 和 C_c 都增 1.

第二阶段开始轮即 $t = 130$ 时令计数器 C_r 的 7 比特重新取值, 由 NFSR 的 6 比特最低有效位串联 LFSR 的 1 比特最低有效位组成, 即 $c_{130}^0 = n_{130}, c_{130}^1 = n_{131}, c_{130}^2 = n_{132}, c_{130}^3 = n_{133}, c_{130}^4 = n_{134}, c_{130}^5 = n_{135}, c_{130}^6 = l_{130}$, 且令 $l_{130} = 1$, 在该阶段 $z_t \oplus v'_t$ 不参与 LFSR 和 NFSR 的反馈, 该阶段跑 80 轮结束整个初始化过程.

则整个的初始化过程为 210(= 130 + 80) 轮, 之后即可输出密钥流 z_t , 在整个初始化阶段是不输出加密密钥的.

3 Cube 攻击

Cube 攻击是由 Dinur 和 Shamir^[1] 在 2009 年的欧密会上提出的一种基于代数思想的选择明文攻击方法, 其目标是恢复密钥. 攻击者把密码算法看作一个黑盒子, 它是关于 n 个未知变量和 m 个公开变量的未知多项式, 并且假设攻击者能得到密码算法的输出比特.

3.1 Cube 攻击原理

将密码算法看做向量空间 F_2^{n+m} 上关于未知变量 k 和公开变量 v 的未知多项式 $P(v, k)$, 其中 $k = (k_1, k_2, \cdots, k_n), v = (v_1, v_2, \cdots, v_m)$, 攻击者可以任意选择公开变量, 并通过询问黑盒得到输出. 假设我们选定公开变量指标的子集 $I = \{i_1, i_2, \cdots, i_l\} \subseteq \{1, 2, \cdots, m\}$, 令 $t_I = v_{i_1}v_{i_2} \cdots v_{i_l}$, 目标多项式分解如下:

$$P(v_1, v_2, \cdots, v_m, k_1, k_2, \cdots, k_n) = t_I P_{S(I)}(\cdot) + Q(v_1, v_2, \cdots, v_m, k_1, k_2, \cdots, k_n) \tag{5}$$

其中, $P_{S(I)}(\cdot)$ 中不含 t_I 中的变量, $Q(v_1, v_2, \cdots, v_m, k_1, k_2, \cdots, k_n)$ 中不含有能被 t_I 整除的项.

称集合 $C_I = \{v_{i_1}v_{i_2}\cdots v_{i_l}\} \subseteq \{v_1, v_2, \dots, v_m\} | i_j \in I, j = 1, 2, \dots, l\}$ 为一个 l 维 Cube, 则 $|C_I| = 2^{|I|}$. 遍历 C_I 所有可能值时, 对式 (5) 求和可得:

$$\sum_{C_I} P(v, k) = \sum_{C_I} t_I P_{S(I)}(\cdot) + \sum_{C_I} Q(v, k) \quad (6)$$

在式 (6) 求和过程中, 对不在 C_I 中的公开变量可以取任意值, 但为了简单考虑一般取 0. 通过式 (6) 得到一个关于 k 和 $v \setminus C_I$ 的多项式 $P_{S(I)}(\cdot)$, 又由不在 C_I 中的公开变量一般取 0, 则 $P_{S(I)}(\cdot)$ 是一个仅关于 k 的多项式. 称 $P_{S(I)}(\cdot)$ 为超级多项式, t_I 为极大项.

Cube 攻击成立的理论依据由下面的等式保证:

$$\sum_{C_I} P(v, k) = P_{S(I)}(\cdot) \bmod 2^{[1]} \quad (7)$$

例 1 $P(k_1, k_2, k_3, v_1, v_2, v_3) = k_1v_1v_2 + k_3v_1v_2 + k_1v_1v_3 + v_1v_2 + k_2 + 1$, 其中 k_1, k_2, k_3 和 v_1, v_2, v_3 分别为未知变量和公开变量, 令 $I = \{1, 2\}$, 则 $P(\cdot)$ 可分解为:

$$P(k_1, k_2, k_3, v_1, v_2, v_3) = v_1v_2(k_1 + k_3 + 1) + k_1v_1v_3 + k_2 + 1$$

Cube 集合 $C_I = \{(0, 0, 0), (0, 1, 0), (1, 0, 0), (1, 1, 0)\}$, Cube 求和可得:

$$\sum_{C_I} P(k_1, k_2, k_3, v_1, v_2, v_3) = P_{S(I)}(\cdot) = k_1 + k_3 + 1$$

3.2 Cube 攻击步骤

基于上面的分析, Cube 攻击分为两个阶段: 预处理阶段和在线加密阶段. 预处理阶段是一个离线加密过程, 攻击者可以随意选取未知变量 k 和公开变量 v 的值并得到相应的输出, 其目标是确定 Cube 使得超级多项式 $P_{S(I)}(\cdot)$ 关于未知变量 k 是线性的. 在线加密阶段攻击者只能选取公开变量 v 的值, 其目标是恢复未知变量 k , 通过在线加密 Cube 的所有可能值并求和, 由式 (7) 可知即为 $P_{S(I)}(\cdot)$ 的值从而得到关于未知变量 k 的线性方程.

预处理阶段: 攻击者随机选择 $c(1 \leq c \leq m)$ 并在 v 中随机选择一个 c 维的子集 C_I , 未选取的 v 中变量设为 0, 遍历 C_I 的所有可能值计算 $P(v, k)$ 并求和得到 $P_{S(I)}(k)$, 判断 $P_{S(I)}(k)$ 是否是线性的, 若是线性的则找到了一个有效的 Cube. 判断 $P_{S(I)}(k)$ 的线性性采用 BLR 方法^[22], 随机选择两个 k 的值 x, y , 若等式 $P_{S(I)}(0) + P_{S(I)}(x) + P_{S(I)}(y) = P_{S(I)}(x + y)$ 成立则可能是线性的, 反之则是非线性的, 这是一个概率性测试方法, 假设测试了 j 次则 $P_{S(I)}(k)$ 是非线性的概率为 2^{-j} . 若 $P_{S(I)}(k)$ 通过了线性测试则需要求出 $P_{S(I)}(k)$ 关于 $k = (k_1, k_2, \dots, k_n)$ 的线性表示, 令 v 和 k 中的所有变量均为 0, 在 C_I 上求 Cube 和即得 $P_{S(I)}(k)$ 的常数项, 令 $k_i (i \in \{1, 2, \dots, n\})$, v 中除了 C_I 中变量和 k 中除了 k_i 的所有变量均为 0, 在 C_I 上求 Cube 和再加上常数项即得 k_i 的系数 (在 GF(2) 上即为 0 或 1), 如此 n 次即可求得 $P_{S(I)}(k)$ 的线性表示. 预处理阶段就是选取不同 v 中变量组合找到足够多能输出关于未知变量的线性多项式的有效 Cube.

在线加密阶段: 攻击者在预处理阶段求得的有效 Cube 的基础上通过在线加密有效 Cube 的所有可能值并将加密结果求和从而得到对应多项式的值, 这就得到了一个线性方程, 通过不同的有效 Cube 求和得到更多的线性方程, 组成线性方程组, 最后求解该方程组即可恢复部分或全部未知变量.

Cube 攻击一般流程见算法1:

算法 1 Cube 攻击算法

Input: 密码算法输出的密钥流比特, Cube 维数, 线性测试次数, 同一维数下 Cube 的寻找次数, 需要搜集的 Cube 数量

Output: 未知变量

```

1  预处理阶段
2  在公开变量中随机选择初始 Cube 的维数大小的变量子集作为初始的待测试 Cube;
3  while 没有达到需要搜集的 Cube 数量 do
4      for 同一维数下 Cube 的寻找次数 do
5          for 线性测试次数 do
6              if 非线性 then
7                  if 该 Cube 维数下 Cube 的寻找次数没有达到要求 then
8                      | 重新选择一个该维数下的 Cube 并作线性测试;
9                  end
10             else
11                 | 增加或减少 Cube 维数;
12             end
13         end
14     else
15         | 计算线性多项式关于未知变量的系数;
16         if 未知变量的所有系数均为 0 then
17             | 重新选择一个 Cube 维数的 Cube 并作线性测试;
18         end
19     else
20         | 输出线性多项式关于未知变量的系数以及对应的 Cube 指标;
21         | 搜集到的 Cube 数量增 1;
22     end
23 end
24 end
25 end
26 end
27 在线加密阶段
28 for 在预处理阶段搜集的 Cube do
29     | 对 Cube 的所有可能值进行在线加密, 并将输出结果求和, 即可得到该 Cube 对应的线性多项式的值从而构建
30     | 方程;
31 end

```

31 将上面得到的所有方程组成方程组, 解此方程组即可恢复未知变量;

4 Fruit 上的 Cube 攻击

用 Cube 攻击分析密码算法的关键问题是如何找到好的有效 Cube, 使得 Cube 的维数尽可能小, 得到线性方程的密码迭代轮数尽可能大, 这是 Cube 攻击的一大难题. 我们通过以往的文献可以发现, 没有太好的理论来获得好的 Cube, 大多数是从密码算法自身的代数结构分析出发来加大得到线性方程的密码迭代轮数. 本文在一般 Cube 攻击理论的基础上提出一些技巧性的想法来找 Cube, 并分析其代数结构, 降低复杂度.

在 Fruit 算法的 Cube 攻击中, 未知变量为密钥 $K = (k_0, k_1, \dots, k_{79})$, 公开变量为初始化向量 $IV = (v_0, v_1, \dots, v_{69})$. 在随机搜索的基础上我们提出如下的一些想法来找有效 Cube:

1. 遍历搜索 6 维以下的 Cube. 这是单机能做到的, 主要目的是看能否发现 Cube 的一些规律性结构, 并在此基础上实施其他寻找 Cube 的想法.

2. 在已找到的有效 Cube 的基础上增加 1 个不在 Cube 中的 IV 变量判断是否也是一个有效 Cube, 同样还可以增加 2 个或 3 个不在 Cube 中的 IV 变量.

3. 在已找到的有效 Cube 中减掉 1 个 Cube 变量判断是否也是一个有效 Cube, 同样也可以减掉 2 个或 3 个 Cube 变量.

4. 将已找到的所有有效 Cube 中的变量形成一个集合, 该集合为 IV 的子集, 在此集合上搜索有效 Cube.

基于上面的这些想法, 我们在单机 Intel(R) Core(TM) i7-4770K CPU @ 3.50GHz 8.00 GB 环境下得到的部分实验结果见表1:

表 1 Fruit 的部分有效 Cube
Table 1 Some good Cubes of Fruit

轮数	Cube 指标	(#)	线性多项式
58	0,9,37,46	(4)*	k_{63}
59	1,10,38,47	(4)	k_{64}
60	2,11,39,48	(4)	k_{65}
...
73	15,24,52,61	(4)	k_{78}
74	16,25,53,62	(4)	k_{79}
73	15,24,52,61	(4) [†]	k_{78}
83	15,24,52,61,64,66,68	(7)	k_{78}
86	15,24,52,61,64,65,66,67,68,69	(10)	k_{78}
84	13,22,50,58,59,62,63,64,65,69	(10) [‡]	k_{76}
82	13,22,50,56,58,59,61,62,63,67	(10)	k_{76}
82	13,22,50,58,59,61,62,63,64,65	(10)	k_{76}

注 1 “*”表示遍历搜索得到的 Cube, “†”表示在 Cube 上增减变量得到新的 Cube, “‡”表示在 Cube 变量形成的子集中找到 Cube.

表1中列举了找到的部分有效 Cube. 第一行有 17 个有效 Cube, 是遍历所有 4 维 Cube 得到的全部, 且 4 维以下找不到任何一个有效 Cube; 第二行是在 Cube(16, 25, 53, 62) 的基础增加 3 个变量得到的新 Cube, 这比在全部变量空间 IV 上搜索 10 维 Cube 的复杂度低得多; 第三行在 Cube(13, 22, 50, 58, 59, 62, 63, 64, 65, 69) 和 (13, 22, 50, 56, 58, 59, 61, 62, 63, 67) 的所有变量形成的子集 {13, 22, 50, 56, 58, 59, 61, 62, 63, 64, 65, 67, 69} 中搜索得到 Cube(13, 22, 50, 58, 59, 61, 62, 63, 64, 65).

从表1的第一行中我们发现 Cube 指标, 轮数和密钥的线性表达式有很好的代数递推关系, Cube 中所有指标都增 1 的情况下, 相对应的得到线性方程的密码迭代轮数也增 1, 且线性多项式中的密钥指标也增 1, 当然 Cube 中所有指标都减 1 也有对应的减 1 递推关系, 只不过对应得到线性方程的密码迭代轮数也减 1 时就不是理想的结果, 所以减 1 的情况可以不用考虑. 如此我们在找到一个有效 Cube 时可以通过其指标都增 1 来判断新的 Cube 是否有效, 这将大大降低搜索 Cube 的复杂度. 这样表1中第三行由 84 轮和 82 轮 Cube 组合得到一个 82 轮的 Cube(13, 22, 50, 58, 59, 61, 62, 63, 64, 65) 并不是一个没有意义的举措, 因为该 Cube 利用递推关系增 1 的情况下最终可以得到 85 轮的 Cube(16, 25, 53, 61, 62, 64, 65, 66, 67, 68), 比参与组合的两个 Cube 的轮数都高.

通过实验我们发现, 上面的递推关系有限制条件, 并不能一直递增下去. Fruit 使用 80 比特密钥 K 和 70 比特初始化向量 IV, 如从 0 开始编号, 则 K 和 IV 的最大指标分别为 79 和 69, 而这也是递推关系的限制条件, 当 Cube 指标达到 69 或线性多项式中的密钥指标达到 79 时递推关系终止, 其指标不能循环即 69 或 79 增 1 变成 0. 而且我们还发现在找到的线性多项式只与 $k_{63}, k_{64}, \dots, k_{79}$ 有关, 没有发现关于 k_0, k_1, \dots, k_{62} 的线性表示.

基于上述的想法和分析, 我们对 83 轮 (最高可到 86 轮) Fruit 做 Cube 攻击求得 17 个密钥, 比穷尽

搜索降低了 2^{17} 的复杂度. 具体结果见表2:

表 2 减轮 Fruit 的 Cube 攻击结果
Table 2 Results of Cube attack on round-reduced Fruit

轮数	Cube 指标	(#)	线性多项式
83	0,9,20,37,46,56,59,61,62,63,64,68	(12)	k_{63}
84	1,10,21,38,47,57,60,62,63,64,65,69	(12)	k_{64}
83	2,11,13,23,33,34,35,39,40,48,59,61,62,69	(14)	k_{65}
83	3,12,23,33,34,35,40,49,59,61,62,69	(12)	k_{66}
83	4,13,24,35,41,50,60,62,63,65,67	(11)	k_{67}
84	5,14,25,36,42,51,61,63,64,66,68	(11)	k_{68}
85	6,15,26,37,43,52,62,64,65,67,69	(11)	k_{69}
84	7,16,44,53,56,58,60,65,67,69	(10)	k_{70}
84	8,17,28,45,54,63,64,67,69	(9)	k_{71}
84	9,18,46,54,55,58,60,65,67,69	(10)	k_{72}
84	10,19,30,47,56,63,65,66,69	(9)	k_{73}
85	11,20,48,57,60,61,62,63,64,65,66,68	(12)	k_{74}
86	12,21,49,58,61,62,63,64,65,66,67,69	(12)	k_{75}
85	13,22,33,50,59,64,66,68,69	(9)	k_{76}
86	14,23,25,51,52,60,64,65,66,67,68,69	(12)	k_{77}
86	15,24,52,61,64,65,66,67,68,69	(10)	k_{78}
85	16,25,37,53,62,64,65,67,69	(9)	k_{79}

5 总结

本文首次尝试了在轻量级流密码 Fruit 上进行 Cube 攻击, 基于一些技巧性想法和代数结构分析的基础上求得 83 轮 (最高可到 86 轮) Fruit 的 17 个密钥. 下一步工作将更深入地分析已有的现象, 以及用条件 Cube 攻击方法研究 Fruit.

References

- [1] DINUR I, SHAMIR A. Cube attacks on tweakable black box polynomials[C]. In: Advances in Cryptology—EUROCRYPT 2009, Springer Berlin Heidelberg, 2009: 278–299.
- [2] LAI X. Higher order derivatives and differential cryptanalysis[J]. KLUWER International Series in Engineering and Computer Science, 1994, 276: 227–233.
- [3] VIELHABER M. Breaking ONE.FIVIUM by AIDA an algebraic IV differential attack[J]. IACR Cryptology ePrint Archive, 2007, 2007: 413.
- [4] De CANNIÈRE C. Trivium: A stream cipher construction inspired by block cipher design principles[C]. In: Information Security—ISC 2006, Springer Berlin Heidelberg, 2006: 171–186.
- [5] AUMASSON J P, DINUR I, MEIER W, et al. Cube testers and key recovery attacks on reduced-round MD6 and Trivium[C]. In: Fast Software Encryption—FSE 2009, Springer Berlin Heidelberg, 2009: 1–22.
- [6] DINUR I, SHAMIR A. Breaking Grain-128 with dynamic Cube attacks[C]. In: Fast Software Encryption—FSE 2011, Springer Berlin Heidelberg, 2011: 167–187.
- [7] MROCZKOWSKI P, SZMIDT J. Cube attack on Courtois toy cipher[J]. IACR Cryptology ePrint Archive, 2009, 2009: 497.

- [8] GHAFARI V A, HU H G, CHEN Y. Fruit-v2: Ultra-lightweight stream cipher with shorter internal state[J]. IACR Cryptology ePrint Archive, 2016, 2016: 355.
- [9] HELL M, JOHANSSON T, MEIER W. Grain: A stream cipher for constrained environments[J]. International Journal of Wireless and Mobile Computing, 2007, 2(1): 86–93.
- [10] WU W, ZHANG L. LBlock: A lightweight block cipher[C]. In: Applied Cryptography and Network Security—ACNS 2011, Springer Berlin Heidelberg, 2011: 327–344.
- [11] BOGDANOV A, KNUDSEN L R, LEANDER G, et al. PRESENT: An ultra-lightweight block cipher[C]. In: Cryptographic Hardware and Embedded Systems—CHES 2007, Springer Berlin Heidelberg, 2007: 450–466.
- [12] GONG Z, NIKOVA S, LAW Y W, KLEIN: A new family of lightweight block ciphers[C]. In: RFID. Security and Privacy—RFIDSec 2011, Springer Berlin Heidelberg, 2011: 1–18.
- [13] BABBAGE S, CANNIERE C, CANTEAUT A, et al. The ECRYPT stream cipher project. eSTREAM. Revised on September 8, 2008.
- [14] BABBAGE S, DODD M. The MICKEY stream ciphers. In: New Stream Cipher Designs—The eSTREAM Finalists, Springer Berlin Heidelberg, 2008: 191–209.
- [15] BIRYUKOV A, SHAMIR A. Cryptanalytic time/memory/data tradeoffs for stream ciphers[C]. In: Advances in Cryptology—ASIACRYPT 2000. Springer Berlin Heidelberg, 2000: 1–13.
- [16] ARMKNECHT F, MIKHALEV V. On lightweight stream ciphers with shorter internal states[C]. In: Fast Software Encryption—FSE 2015, Springer Berlin Heidelberg, 2015: 451–470.
- [17] LALLEMAND V, PLASENCIA M N. Cryptanalysis of full Sprout[C]. In: Advances in Cryptology—CRYPTO 2015, Springer Berlin Heidelberg, 2015: 663–682.
- [18] ESGIN M F, KARA O. Practical cryptanalysis of full Sprout with TMD tradeoff attacks[C]. In: Selected Areas in Cryptography—SAC 2015, Springer Cham, 2015: 67–85.
- [19] ZHANG B, GONG X. Another tradeoff attack on sprout-like stream ciphers[C]. In: Advances in Cryptology—ASIACRYPT 2015, Springer Berlin Heidelberg, 2015: 561–585.
- [20] HAMANN M, KRAUSE M, MEIER W. LIZARD—A lightweight stream cipher for power constrained devices[J]. IACR Cryptology ePrint Archive, 2016, 2016: 926.
- [21] MIKHALEV V, ARMKNECHT F, MÜLLER C. On ciphers that continuously access the non-volatile key[J]. IACR Transactions on Symmetric Cryptology, 2016, 2016(2): 52–79.
- [22] BLUM M, LUBY M, RUBINFELD R. Self-testing/correcting with applications to numerical problems[J]. Journal of Computer and System Sciences, 1993, 47(3): 549–595.

作者信息



孙移盛 (1980–), 安徽望江人, 硕士. 主要研究领域为流密码算法的分析.

E-mail: sun-ys14@mails.tsinghua.edu.cn