密码学报 ISSN 2095-7025 CN 10-1195/TN
Journal of Cryptologic Research,2016,3(3):258–269

E-mail: jcr@cacrnet.org.cn
http://www.jcr.cacrnet.org.cn
Tel/Fax: +86-10-81033101

Grain-v1 的多比特差分故障攻击*

叶晨东[1,2], 田甜[1]

1. 信息工程大学, 郑州 450001
2. 密码科学技术国家重点实验室, 北京 100878
通讯作者: 田甜, E-mail: tiantian_d@126.com

摘 要: 本文研究 Grain-v1 的差分故障攻击. 目前, 很多文献在一个故障引起一个中间状态比特翻转的假设条件下, 利用差分故障攻击对 Grain 系列算法进行了分析. 然而, 随着芯片尺寸的缩小以及复杂性的提升, 一个故障精确地引起一个中间状态比特的翻转在技术上实现的难度越来越大. 对于 Grain-v1, 目前并没有文献在一个故障引起多个中间状态比特翻转的假设条件下, 给出一个有效的差分故障攻击. 本文针对 Grain-v1, 在一个故障至多引发连续 8 比特翻转, 翻转比特的位置可以是 LFSR, 或者 NFSR, 或者横跨 LFSR 和 NFSR, 并且具体翻转比特数量未知的条件下, 给出了一个有效的差分故障攻击. 特别地, 文中利用在 FSE 2013 中提出的 Grain-v1 近似碰撞攻击的思想, 给出了一个新的确定故障信息的方法, 即故障实际引发的比特翻转位置和比特翻转数量. 实验数据表明, 已知 160 比特的差分序列, 该方法能以大约 97.5% 的概率确定出故障信息. 通过 SAT 求解器 CryptoMiniSat2.9.6, 在 CPU 频率为 2.83GHz、4G 系统内存的 PC 机上, 利用大约 8 个故障, 五十分钟左右可以恢复出 Grain-v1 的 160 比特中间状态. 本文攻击思想也适用于 Grain-128 以及一个故障引发大于 8 比特翻转的情形.
关键词: 序列密码; Grain-v1; 差分故障攻击
中图法分类号: TP309.7 **文献标识码**: A DOI: 10.13868/j.cnki.jcr.000126

中文引用格式: 叶晨东, 田甜. Grain-v1 多比特情形下的差分故障攻击[J]. 密码学报, 2016, 3(3): 258–269.
英文引用格式: YE C D, TIAN T. Multi-bit differential fault analysis of Grian-v1[J]. Journal of Cryptologic Research, 2016, 3(3): 258–269.

Multi-Bit Differential Fault Analysis of Grain-v1

YE Chen-Dong[1,2], TIAN Tian[1]

1. Information Engineering University, Zhengzhou 450001, China
2. State Key Laboratory of Cryptology, Beijing 100878, China
Corresponding author: TIAN Tian, E-mail: tiantian_d@126.com

Abstract: This paper studies differential fault attack against Grain-v1. Recently several differential fault attacks were reported on Grain family under the assumption that a single fault could flip a single bit of the internal state. However, as chip sizes shrink and the complexity of devices increases, one bit of internal state being flipped by a single fault with acceptable accuracy seems to be more and more difficult in practice. As for Grain-v1, no efficient multi-bit differential fault attack has been proposed yet. This paper presents a multi-bit differential attack against Grain-v1, under the assumption that a single fault could flip no more than 8 consecutive bits in the main

* 基金项目: 国家自然科学基金(61272042, 61521003); 国家 863 重点项目(2015AA01A708)
收稿日期: 2015-09-29 定稿日期: 2016-03-08

register without knowing the specific location and the exact number of bits. Those flipped bits could be located at the LFSR, or at the NFSR, or even across the LFSR and the NFSR. In particular, inspired by the main idea of near collision attack against Grain-v1 proposed in FSE 2013, a new method of identifying a multi-bit fault is proposed, including the position and the number of the flipped bits. By this new method, using 160 differential key-stream bits, the corresponding fault information could be determined with a probability of 97.5%. By the SAT solver CryptoMiniSat2.9.6, on a computer with a 2.83GHz CPU and 4G RAM, the 160-bit internal state of Grain-v1 could be recovered within 50 minutes using about eight faults. The idea of the analysis in this paper could also be applied to Grain-128 and the case of more than 8 bits flipped by a single fault.
Key words: stream ciphers; Grain-v1; differential fault attack

1 引言

Grain-v1 算法[1]是一种面向硬件实现的轻量级序列密码算法, 由 Hell、Johansson 和 Meier 于 2005 年提出, 并且入选了 eStream 计划最终的推荐算法. Grain-v1 算法支持 80 比特私钥(Key)和 64 比特初始化向量(IV), 算法结构十分紧凑: 主要由一个 80 级的线性反馈移位寄存器、一个 80 级的非线性反馈移位寄存器、一个 5 元 3 次前馈函数 h 构成. 随后, 完全基于 Grain-v1 的设计思想, 设计者又分别提出了支持 128 比特私钥的 Grain-128 算法 268[2]和支持认证功能的 Grain-128a 算法[3]. Grain 系列算法提出后, 受到密码分析者的大量关注, 出现了许多有意义的攻击方法和结果, 例如滑动相关密钥攻击[4]、动态 Cube 攻击[5]、条件差分攻击[6]、近似碰撞攻击[7][7] 、差分故障攻击[8–13]等. 本文主要关注差分故障攻击.

故障攻击, 是一种侧信道攻击方法, 最初由 Boneh、Demillo 和 Lipton 在 1996 年提出, 用于攻击 RSA 等基于数论的公钥密码[14]. 之后 Biham 和 Shamir 利用该方法实现了对 DES 等分组密码的攻击[15]. 虽然故障攻击被广泛的应用于公钥密码和分组密码的攻击上, 但是直到 2004 才由 Jonathan 和 Shamir 将故障攻击应用到对序列密码的分析上[16]. 差分故障攻击, 结合了故障攻击和差分攻击, 允许攻击者在加密过程中引入故障使得加密算法的中间状态发生变化, 然后通过分析正确密钥流和引入故障后产生的错误密钥流, 可以得出部分或者完整的中间状态的信息, 进而恢复出密钥的信息. 下面重点介绍 Grain 系列算法的差分故障攻击研究结果.

文献[8,9]首先研究了 Grain-128 的差分故障攻击. 文献[8]假设攻击者能够在同一位置反复引入故障, 并且故障只能引入在线性反馈移位寄存器(LFSR)上. 另外, 文献[8]中还假设攻击者能够控制故障的引入时间并能在保持私钥和 IV 不变的前提下重置加密机. 基于上述假设, 利用连续 24 个故障, 文献[8]攻击方法在几分钟之内可恢复出私钥. 文献[9]增加了攻击者在保持私钥不变的同时能采用不同的 IV 启动加密机的假设条件, 并将故障的引入位置扩展到非线性反馈移位寄存器(NFSR)上. 利用文献[9]中给出的方法, 攻击者引入 56 个故障可以恢复出 NFSR 的状态, 引入 128 个故障可以恢复出 LFSR 的状态. 由于文献[8]和文献[9]的攻击方法都利用了 Grain-128 算法中 h 函数的特殊性质, 并不能简单地将其方法直接应用到 Grain-v1 算法的攻击上.

2012 年, 在与文献[8]相同的假设条件下, 文献[10]提出了通过寻找合适的 α 使得 $h(x)+h(x+\alpha)$ 是一个线性函数的方法, 首次将差分故障攻击应用于 Grain-v1 算法. 并且, 文献[10]不限制故障引入的位置, 即故障既可以引入在 LFSR 上也可以引入在 NFSR 上. 但是, 文献[10]中攻击所需要的故障数量非常大, 约为 530 次故障, 很大程度上限制了攻击的实用性.

文献[8–10]都假设了攻击者可以在同一个位置反复引入故障. 针对这一点, 文献[11]进行了改进, 攻击不再需要该假设条件. 并且当故障引入在 LFSR 上时, 文献[11]允许一个故障可以改变 LFSR 上至多 3 个连续比特. 但是, 文献[11]中的方法只能区别故障是否引起一比特中间状态的翻转, 对于引起多比特翻转的故障, 并不能具体确定故障的位置和翻转的比特数. 另外, 文献[11]中攻击所需要的故障数量仍然较大. 文

献[12]在文献[11]的基础上进一步减弱了攻击的假设条件, 即允许故障引入的时间和位置都是未知的. 另外, 文献[12]中引入 SAT 求解器, 将攻击 Grain-v1 算法、Grain-128 算法、Grain-128a 算法所需要的故障数量分别降低到了 10、4 和 10. 此外, 当故障引入在 NFSR 上时, 文献[12]中方法能有效区分出该故障是否引发单个中间状态比特的翻转. 文献[13]针对 Grain-128 在一个故障至多引起连续 5 个中间状态比特翻转的假设下, 结合 SAT 求解器给出了一个差分故障攻击方法, 并且只需要引入 5 个故障就可以恢复出算法中间状态. 而对于一个故障引起多比特翻转的情形, 目前还没有文献针对 Grain-v1 给出可行的差分故障攻击.

实现差分故障攻击的一个关键就是在加密装置中引入故障, 很多文献对其进行了研究. 目前用于引入故障的物理手段一般有以下几种: 电压瞬变、外部时钟突变、改变电路环境的温度、激光束、光学手段以及 X-射线和离子束[17]. 随着芯片尺寸变小以及复杂性提升, 单个故障以足够高的精度改变加密过程中间状态某一比特的状态在技术上实现的难度将越来越大[18].

本文研究了在一个故障引发连续多比特翻转的情形(简称多比特情形)下, Grain-v1 的差分故障攻击. 特别地, 利用近似碰撞攻击的思想, 本文给出了一个新的确定差分信息的方法. 结合 SAT 求解器, 只需要 8 个故障以及 160 比特输出密钥流, 攻击者在五十分钟之内可以恢复出 Grain-v1 的中间状态.

本文结构安排如下: 第 2 节对 Grain-v1 进行了简要介绍并且给出了差分故障攻击的一些准备知识. 具体攻击过程将在第 3 节中给出. 第 4 节, 本文针对 Grain-v1 进行了实验, 并给出了具体的实验结果.

2 准备工作

本节对 Grain-v1 进行了简单介绍并且给出了差分故障攻击的准备工作.

2.1 Grain-v1 算法简介

Grain-v1 的密钥长度是 80 比特, IV 是 64 比特, 中间状态为 160 比特. 算法主要包括三个组件: 一个 80 比特的 LFSR, 一个 80 比特的 NFSR 以及一个非线性过滤函数 h.

记 t 时刻 LFSR 和 NFSR 的状态分别为 $Y_t=(y_t,y_{t+1},\cdots,y_{t+79})$ 和 $X_t=(x_t,x_{t+1},\cdots,x_{t+79})$. 对于 $t\geqslant 0$, LFSR 的更新函数为

$$y_{n+t}=f(Y_t)=y_t\oplus y_{t+13}\oplus y_{t+23}\oplus y_{t+38}\oplus y_{t+51}\oplus y_{t+62}$$

NFSR 的更新函数为

$$x_{n+t}=y_t\oplus g(X_t)$$

其中函数 g 是 $\mathbb{F}_2$ 上的一个 6 次函数, 其具体代数表达式为

$$\begin{aligned}g(X_t)=&x_{t+62}\oplus x_{t+60}\oplus x_{t+52}\oplus x_{t+45}\oplus x_{t+37}\oplus x_{t+33}\oplus x_{t+28}\oplus x_{t+21}\\&\oplus x_{t+14}\oplus x_{t+9}\oplus x_t\oplus x_{t+63}x_{t+60}\oplus x_{t+37}x_{t+33}\oplus x_{t+15}x_{t+9}\\&\oplus x_{t+60}x_{t+52}x_{t+45}\oplus x_{t+33}x_{t+28}x_{t+21}\oplus x_{t+63}x_{t+45}x_{t+28}x_{t+9}\\&\oplus x_{t+60}x_{t+52}x_{t+37}x_{t+33}\oplus x_{t+63}x_{t+60}x_{t+21}x_{t+15}\\&\oplus x_{t+63}x_{t+60}x_{t+52}x_{t+45}x_{t+37}\oplus x_{t+33}x_{t+28}x_{t+21}x_{t+15}x_{t+9}\\&\oplus x_{t+60}x_{t+52}x_{t+45}x_{t+37}x_{t+33}x_{t+28}x_{t+21}\end{aligned}$$

非线性过滤函数定义为

$$\begin{aligned}h(X_t,Y_t)=&h(y_{t+3},y_{t+25},y_{t+46},y_{t+64},x_{t+63})\\=&y_{t+25}\oplus x_{t+63}\oplus y_{t+3}y_{t+64}\oplus y_{t+46}y_{t+64}\\&\oplus y_{t+64}x_{t+63}\oplus y_{t+3}y_{t+25}y_{t+46}\\&\oplus y_{t+3}y_{t+46}y_{t+64}\oplus y_{t+3}y_{t+46}x_{t+63}\\&\oplus y_{t+25}y_{t+46}x_{t+63}\oplus y_{t+25}y_{t+64}x_{t+63}\end{aligned}$$

Grain-v1 的输出密钥流通过结合 LFSR 和 NFSR 状态得到, 即

$$z_t = h_1(X_t, Y_t) = \oplus_{a\in A} x_{t+a} \oplus h(X_t, Y_t)$$

其中 $A=\{1,2,4,10,31,43,56\}$.

Grain-v1 算法运行过程主要分以下两个阶段:

(1) Key/IV 初始化算法

首先用 80 比特密钥填充 NFSR, 然后用 64 比特 IV 填充 LFSR, 不足部分全部补 1. 系统空跑 160 拍, 并将算法输出 z_t 分别反馈回 LFSR 和 NFSR, 如图 1(包含虚线部分).

(2) 密钥流生成算法(PRGA)

在这个阶段中, z_t 不再反馈给 LFSR 和 NFSR, 而是直接输出作为密钥流, 如图 1(不包含虚线部分).

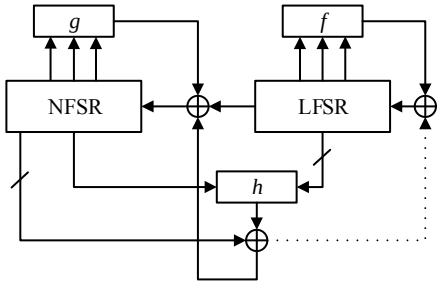

图 1 Grain-v1 算法示意图
Figure 1 Structure of Grain-v1

2.2 故障差分攻击准备工作

记 Grain-v1 算法 t 刻中间状态为 $S_t = X_t \| Y_t = (x_t, x_{t+1}, \cdots, x_{t+79}, y_t, y_{t+1}, \cdots, y_{t+79}) = (s_{t+0}, \cdots, s_{t+159})$, $t\geqslant 0$, 其中 S_0 是 Grain-v1 的 PRGA 阶段的初始状态. 那么, 一个故障 $\phi_{(k,m)}$, 指的是在初始状态 S_0 中引起了第 $k+1$ 比特到第 $k+m$ 比特的翻转, 即

$$S_{0,\Delta\phi_{(k,m)}} = (s_0, \cdots, s_{k-1}, s_k \oplus 1, \cdots, s_{k+m-1} \oplus 1, \cdots, s_{159})$$

是引入故障之后的初始状态.

记 $Z=[z_0,\cdots,z_{l-1}]$ 和 $Z^{\phi}=[z_0^{\phi}, z_1^{\phi}, \cdots, z_{l-1}^{\phi}]$ 分别表示由 S_0 和 $S_{0,\Delta\phi_{(k,m)}}$ 产生的前 l 比特输出序列. 若将 $s_0, s_1, \cdots, s_{159}$ 看成变量, 对于某个 $\phi_{(k,m)}$, 利用符号计算可以得出 z_i 和 z_i^{ϕ} 具体的代数表达式. 将 Z 和 Z^{ϕ} 的差分记为 E_{ϕ}, 即

$$E_{\phi} = (z_0 \oplus z_0^{\phi}, z_1 \oplus z_1^{\phi}, \cdots, z_{l-1} \oplus z_{l-1}^{\phi})$$

设 $1\leqslant i\leqslant l-1$, 如果 $E_{\phi}(i)=1$, 则说明对于任意的初始状态 S_0, 差分 $z_i \oplus z_i^{\phi}$ 都等于 1; 如果 $E_{\phi}(i)=0$, 则说明对于任意的初始状态 S_0, 差分 $z_i \oplus z_i^{\phi}$ 都等于 0; 若 $E_{\phi}(i)\neq 0$ 且 $E_{\phi}(i)\neq 1$, 则说明随着初始状态 S_0 的变化, 差分 $z_i \oplus z_i^{\phi}$ 也在变化, 不是固定值. 记 $\mathrm{Sig}_{\phi}^{1}=\{i \mid E_{\phi}(i)=1\}$ 和 $\mathrm{Sig}_{\phi}^{0}=\{i \mid E_{\phi}(i)=0\}$. 此外, 对于某些 i 和 j, $E_{\phi}(i)$ 和 $E_{\phi}(j)$ 都不等于 0 和 1, 但是 $E_{\phi}(i) \oplus E_{\phi}(j)=1$ 或 0, 将这些点对 (i,j) 的集合分别记成 $\mathrm{Sig}_{\phi}^{\neq}$ 和 $\mathrm{Sig}_{\phi}^{=}$. 此后,

将 Sig_{ϕ}^{1}, Sig_{ϕ}^{0}, $\mathrm{Sig}_{\phi}^{\neq}$ 和 $\mathrm{Sig}_{\phi}^{=}$ 统称为标签向量. 通过分析标签向量可以对不同故障的 $\phi_{(k,m)}$ 进行区分.

3 攻击假设条件和方法描述

本节给出对 Grain-v1 的多比特差分故障攻击. 首先, 给出本文差分故障攻击的假设条件:

(1) 加密机可以重置, 即可以保持相同的 Key 和 IV 重新启动加密机.

(2) 假设 PRGA 阶段的 0 时刻引入故障.

(3) 一个故障能够引起至多连续 8 个中间状态比特的翻转.

(4) 故障既可以引入在 LFSR 上, 也可以引入在 NFSR 上, 甚至故障引起翻转的比特可以横跨 LFSR 和 NSFR.

其次, 介绍攻击的整个过程. 第一步, 在PRGA阶段引入故障, 用给定算法逐个确定的故障的信息, 包括故障的位置和翻转的比特数; 第二步, 根据故障的信息生成关于 PRGA 阶段初态的方程组; 第三步, 重复上述两个步骤若干次以获得足够的方程, 利用 SAT 求解器求解 PRGA 阶段的初态; 第四步, 根据 PRGA 阶段的初态和 Grain-v1 算法的可逆性来恢复私钥.

3.1 改进的D-Grain算法

由于 Grain-v1 算法中非线性反馈函数的次数比较高, 直接利用符号计算 z_l 并不可行. 例如, 当 $l>44$ 时, 在一台 1.83 GHz 处理器, 3 GHz RAM, 2 MB 系统缓存的机器上计算 Grain-v1 中 z_l 的代数表达式已经较为困难[12].

文献[8]中首次给出 ΔGrain 算法, 并且利用 ΔGrain 算法来确定故障的位置. 文献[11]对 ΔGrain 算法进行了改进, 给出了 D-Grain(Differential Grain)算法来确定故障的信息. 本文将对 D-Grain 算法进行简单改进.

D-Grain 算法包括一个 80 比特 LFSR 和一个 80 比特 NFSR, 分别记为 ΔL 和 ΔN, 并将它们 t 时刻的状态分别记为 $\Delta L_t=[u_t,\cdots,u_{t+79}]$ 和 $\Delta N_t=[v_t,\cdots,v_{t+79}]$. ΔL 和 ΔN 的初态中除了 S_0 和 $S_{0,\Delta\phi_{(k,m)}}$ 差分为 1 的位置等于 1, 其余位置都设置成 0.

ΔL 的更新函数与 Grain-v1 中 LFSR 的反馈函数一致. 对于 ΔN 的反馈函数, 文献[11]中定义为

$$v_{t+80}=u_t+2\cdot \mathrm{OR}\left(v_t,v_{t+\tau_1},\cdots,v_{t+\tau_b}\right)$$

其中 OR 运算定义如下:

$$\mathrm{OR}\left(v_t,v_{t+\tau_1},\cdots,v_{t+\tau_b}\right)=\begin{cases}0, & v_t=v_{t+\tau_1}=\cdots=v_{t+\tau_b}=0\\ 1, & \text{其它}\end{cases}$$

而 $v_t,v_{t+\tau_1},\cdots,v_{t+\tau_b}$ 是 Grain-v1 的 NFSR 反馈函数 g 的抽头.

注意到 ΔL 和 ΔN 的初态 ΔL_0 和 ΔN_0 实际上给出了 S_0 和 $S_{0,\Delta\phi_{(k,m)}}$ 的差分. 由 ΔL 和 ΔN 更新函数的定义, ΔL_t 和 ΔN_t 则在一定程度上说明了 S_t 和 $S_{t,\Delta\phi_{(k,m)}}$ 之间的差分情况. 如果 ΔL_t 和 ΔN_t 中的某一位置为 0, 那么 S_t 和 $S_{t,\Delta\phi_{(k,m)}}$ 中对应位置的两个比特以概率 1 相等; 如果 ΔL_t 和 ΔN_t 中的某一位置为 1, 那么 S_t 和 $S_{t,\Delta\phi_{(k,m)}}$ 中对应位置的两个比特以概率 1 不相等; 如果 ΔL_t 和 ΔN_t 中的某一位置为 2 或者 3, 那么 S_t 和 $S_{t,\Delta\phi_{(k,m)}}$ 中对应位置的两个比特以概率 p 不相等, $0<p<1$. 因此, 如果能够增加 ΔL_t 和 ΔN_t 中 0 或 1 的数量, 就能够更加精确地反应 S_t 和 $S_{t,\Delta\phi_{(k,m)}}$ 的差分情况.

实际上, 在 Grain-v1 算法中 NFSR 的更新函数中有 4 个抽头(LFSR 一个抽头 y_t, NFSR 三个抽头 x_t, x_{t+14}, x_{t+62})只在线性部分出现, 而上述 ΔN 的反馈函数并没有将更新函数中线性抽头和非线性抽头区分对待. 这使得某些时刻的 v_{t+80} 应该被设置成 0 或 1, 却被设置成 2. 现在给出新的反馈函数的定义为:

$$v_{t+80}=\begin{cases}u_t, & v_t\oplus v_{t+14}\oplus v_{t+62}=0\text{且}v_{t+\tau_i}=0,\tau_i\notin\{0,14,62\}\\ u_t\oplus 1, & v_t\oplus v_{t+14}\oplus v_{t+62}=1\text{且}v_{t+\tau_i}=0,\tau_i\notin\{0,14,62\}\\ 2, & \text{其它}\end{cases}$$

在定义 D-Grain 算法的输出之前, 说明以下符号的含义.

(1) χ_t 表示 t 时刻 NFSR 中出现在计算输出公式线性部分的比特的集合, 即 $\chi_t=\{v_{t+1},v_{t+2},v_{t+4},v_{t+10},v_{t+31},v_{t+43},v_{t+56}\}$;

(2) γ_t 表示 t 时刻 Grain-v1 算法中 h 函数中来自 LFSR 和 NFSR 的抽头;

(3) $B\prec\beta$ 表示向量 B 中的每一个分量的绝对值都小于正整数 β;

(4) $\|A\|$ 表示集合 A 中非零元素的个数.

于是, D-Grain 算法的输出定义如下

$$\Delta_{z_t}=\begin{cases}0, & \gamma_t=\mathbf{0},\chi_t\prec 1,\text{且}\|\chi_t\|\text{是偶数}\\ 1, & \gamma_t=\mathbf{0},\chi_t\prec 1,\text{且}\|\chi_t\|\text{是奇数}\\ 2, & \text{其它}\end{cases}$$

其中 $\mathbf{0}$ 表示全零向量.

如果 $\gamma_t=\mathbf{0}$, 则说明 S_t 和 $S_{t,\Delta\phi_{(k,m)}}$ 中非线性函数 h 的输入比特都是相等的(注: 对任意的 S_0 都成立). 如果 χ_t 中所有元素都小于等于 1, 说明 S_t 和 $S_{t,\Delta\phi_{(k,m)}}$ 中出现在输出函数的线性部分中的比特以概率 1 相等或者不等. 当上述的事件同时发生时, 如果 $\|\chi_t\|$ 是偶数, 那么由 S_t 和 $S_{t,\Delta\phi_{(k,m)}}$ 产生的输出比特以概率 1 相等, 此时 D-Grain 算法输出为 0; 如果 $\|\chi_t\|$ 是奇数, 那么由 S_t 和 $S_{t,\Delta\phi_{(k,m)}}$ 产生的输出比特以概率 1 不相等, 此时 D-Grain 算法输出为 0. 其它情况下, D-Grain 算法输出为 2.

从上述分析中可以看出, 如果将 D-Grain 算法运行 r 拍, 可以得到分别由 S_t 和 $S_{t,\Delta\phi_{(k,m)}}$ 产生的前 r 比特中以概率 1 相等或者不相等的位置, 即密钥流中确定性差分的位置. 另外, 可以观察到, $\Delta N_t=[v_t,\cdots,v_{t+79}]$ 中 0 和 1 的数量越多, $\mathrm{Sig}^1_{\phi(k,m)}$ 和 $\mathrm{Sig}^0_{\phi(k,m)}$ 中的元素越多, 从而故障 $\phi(k,m)$ 更容易识别.

3.2 确定故障信息

由于本文假设在 PRGA 阶段 0 时刻引入故障, 所以对于一个故障, 只有其位置以及翻转比特数是未知的. 因而, 一个故障可能引发 $\sum_{m=1}^{8}(161-m)=1252$ 种中间状态翻转情形, 记这些情形的集合为 Γ. 本节的主要工作是对一段给定的差分序列, 唯一得确定其对应故障的位置和翻转比特数量, 即确定故障信息.

文献[13]中针对 Grain-128 考虑了至多连续 5 比特的差分情形, 但是 Grain-v1 的非线性反馈函数的次数为 6, 比 Grain-128 的反馈函数更为复杂. 因此直接利用文献[13]中用于确定故障信息的方法, 并不能有效地确定本文模型中故障的信息. 实验表明(参见第 3.2.1 节), 对所有的 $\phi_{(k,m)}\in\Gamma$, 生成正确密钥流和故障密钥流之间的差分序列, 然后利用文献[13]中的方法来确定 $\phi_{(k,m)}$ 的信息, 结果能够唯一确定故障的比例非常

低. 因此, 本文结合以下两种手段来确定故障信息:

(1) 利用标签向量确定故障信息, 参见文献[13];

(2) 利用近似碰撞攻击(near collision attack, NCA)确定故障信息.

3.2.1 利用标签向量确定故障信息

在实验中我们取 $r=160$, 对每一个 $\phi_{(k,m)}\in\Gamma$, 利用 D-Grain 算法预计算得到前 160 轮的 $\mathrm{Sig}^1_{\phi_{(k,m)}}$ 和 $\mathrm{Sig}^0_{\phi_{(k,m)}}$. 对于一段给定的差分序列 ΔZ, 利用算法 1 给出所有可能得到这段差分序列的故障, 算法 1 描述见表 1, 其中的 $\mathrm{support}^0(\Delta Z)$ 表示序列 ΔZ 中等于 0 的位置的集合, $\mathrm{support}^1(\Delta Z)$ 表示序列 ΔZ 中等于 1 的位置的集合.

表 1 利用 $\mathrm{Sig}^1_{\phi(k,m)}$ 和 $\mathrm{Sig}^0_{\phi(k,m)}$ 确定故障信息算法流程

Table 1 The Algorithm of identifying the information of faults with $\mathrm{Sig}^1_{\phi(k,m)}$ and $\mathrm{Sig}^0_{\phi(k,m)}$

算法 1: Identify_Fault_Information
输入: ΔZ
输出: 所有可能的故障集合 P

```
Identify-Fault-Information( ΔZ )
{
    P ← ∅
    for all  φ∈Γ  do
        if (PossibleFault (ΔZ,φ) ==true)
            P ← P∪(k,m)
    end for
    return  P
}
PossibleFault (ΔZ,φ(k,m))
{
    if( Sig^0_φ(k,m) ⊆ support^0(ΔZ) )
        if( Sig^1_φ(k,m) ⊆ support^1(ΔZ) ) return true
        else return false
    else return false
}
```

为了检测算法 1 的效果, 对每一个 $\phi_{(k,m)}\in\Gamma$ 随机选择 Key 和 IV, 并生成对应的差分序列, 然后利用算法 1 确定故障的信息. 实验结果表明, 算法 1 的成功概率不足 25%. 最终只有 301 个 $\phi_{(k,m)}\in\Gamma$, 能由差分序列唯一确定 $\phi_{(k,m)}$ 的信息, 即由差分序列唯一确定一个二元组 (k,m). 因此, 可以认为只利用 $\mathrm{Sig}^0_{\phi_{(k,m)}}$ 和 $\mathrm{Sig}^1_{\phi_{(k,m)}}$ 很难唯一地确定故障的信息, 故考虑使用 $\mathrm{Sig}^=_{\phi_{(k,m)}}$ 和 $\mathrm{Sig}^{\neq}_{\phi_{(k,m)}}$ 来改进算法.

文献[9]第一次在 Grain 系列算法的差分故障分析中使用了 $\mathrm{Sig}^=_{\phi_{(k,m)}}$ 和 $\mathrm{Sig}^{\neq}_{\phi_{(k,m)}}$. 利用类似于文献[9]中的方法, 我们试图对于所有的 $\phi_{(k,m)}\in\Gamma$ 计算出对应的 $\mathrm{Sig}^=_{\phi_{(k,m)}}$ 和 $\mathrm{Sig}^{\neq}_{\phi_{(k,m)}}$. 实验结果表明, 当 m 越来越大, $\mathrm{Sig}^=_{\phi_{(k,m)}}$ 和 $\mathrm{Sig}^{\neq}_{\phi_{(k,m)}}$ 的数量越来越少, 甚至大部分 $\phi_{(k,m)}$ 不存在 $\mathrm{Sig}^=_{\phi_{(k,m)}}$ 和 $\mathrm{Sig}^{\neq}_{\phi_{(k,m)}}$. 具体结果见表, 其中 $k(i,j)$ 表示故障引入在 k 处时, (i,j) 属于 $\mathrm{Sig}^=_{\phi_{(k,m)}}$ 和 $\mathrm{Sig}^{\neq}_{\phi_{(k,m)}}$.

从表 2 中可以看出当 $6\leqslant m\leqslant 8$ 时, $\mathrm{Sig}^{\neq}_{\phi_{(k,m)}}$ 中元素非常少, 甚至 $\mathrm{Sig}^=_{\phi_{(k,m)}}$ 是空集. 因此可以认为, 在多比特情形下, 对于 Grain-v1 算法的差分故障攻击, $\mathrm{Sig}^=_{\phi_{(k,m)}}$ 或 $\mathrm{Sig}^{\neq}_{\phi_{(k,m)}}$ 对于确定故障信息的帮助不大.

表 2 $\mathrm{Sig}^{=}_{\phi_{(k,m)}}$ 和 $\mathrm{Sig}^{\neq}_{\phi_{(k,m)}}$ $(m\in[5,8])$

Table 2 $\mathrm{Sig}^{=}_{\phi_{(k,m)}}$ and $\mathrm{Sig}^{\neq}_{\phi_{(k,m)}}$ $(m\in[5,8])$

m	$\mathrm{Sig}^{=}_{\phi_{(k,m)}}$	$\mathrm{Sig}^{\neq}_{\phi_{(k,m)}}$
8	无	8(24,37), 9(25,38), 10(26,39), 11(27,40)
7	无	8(24,37), 9(24,37),(25,38), 10(25,38),(26,39), 11(26,39),(27,40)
6	无	8(24,37), 9(24,37),(25,38), 10(24,37),(25,38),(26,39) 11 (25,38),(26,39),(27,40)
5	0(44,57) ,62(26,39),77(41,54) 1(45,58) ,63(27,40),78(42,55) 2(46,59) ,64(28,41),79(43,56) 3(47,60) ,65(29,42),80(44,57) 4(48,61) ,66(30,43),81(45,58) 8(24,37),67(31,44),82(46,59) 9(25,38),68(32,45),83(47,60) 10(26,39),69(33,46),84(48,61) 11(27,40),70(34,47),85(49,62) 29(24,37),71(35,48),86(50,63) 30(25,38),72(36,49),87(51,64) 31(26,39),73(37,50),88(52,65) 32(27,40),74(38,51),89(53,66) 60(24,37),75(39,52),90(54,67) 61(25,38),76(40,53)	9 (24,37),(24,49) 10 (25,38),(25,50) 11 (25,38),(26,39),(26,51) 12(25,38),(27,40) 13(26,39), 14(27,40)

3.2.2 利用 NCA 方法确定故障信息

2013 年, 文献[7]中给出了一种密钥恢复攻击方法, 称为近似碰撞攻击. 文献[7]的作者发现对于低重中间状态差分, Grain-v1 的密钥流差分分布非常不均衡, 从而根据密钥流差分可以以一定概率猜测中间状态差分. 注意到这一点与故障信息的确定有相似性. 于是本文结合 NCA 攻击的思想给出一种新的确定故障信息的方法.

设 P 为算法 1 的输出, 如果 $|P|=1$, 则直接输出 P. 如果 $|P|\neq 1$, 对每一个 $\phi_{(k,m)}\in P$, 随机生成 2^{12} 个内部状态 S_i, $1\leqslant i\leqslant 2^{12}$, 然后翻转 S_i 的第 k 到第 $k+m-1$ 比特, 记为 S_i', 从而获得符合 $\phi_{(k,m)}$ 定义的 2^{12} 对中间状态 (S_i,S_i'). 对 $1\leqslant i\leqslant 2^{12}$, 分别利用 S_i 和 S_i' 生成长度为 30 的输出序列段, 这些序列段差分的集合记为 ΔH. 统计 ΔH 中各元素的出现频率, 记录在一个表中, 以 (k,m) 命名. 取差分序列 ΔZ 的前 30 比特 H, 遍历所有生成的表, 返回 H 出现频率最高的表的文件名. 本文认为返回的文件名正是所要确定的故障信息. 具体算法见表 3.

3.3 生成方程组

本节将说明如何利用正确密钥流以及故障密钥流来生成关于 GRPA 初态的方程组.

3.3.1 利用正确密钥流生成方程组

考虑正确密钥流 $(z_0,\cdots,z_{l-1})$, 并且增加新的变元 x_{t+80},y_{t+80} ($0\leqslant t\leqslant l-1$)以控制得到的方程的代数次数. 于是, 可以得到以下几类方程:

(1) 线性反馈方程: $y_{t+80}=f(Y_t)$;

(2) 非线性反馈方程: $x_{t+80}=y_{t+80}\oplus g(X_t)$;

(3) 密钥序列方程: $z_t=\oplus_{a\in A}x_{t+a}\oplus h(X_t,Y_t)$;

通过上述方法, 利用 l 比特长的密钥流 $(z_0,\cdots,z_{l-1})$, 一共可以得到 $3l$ 个方程, 增加了 $2l$ 个变元. 最终可以得到一个含有 $3l$ 个方程, $160+2l$ 个变元的非线性方程组.

表 3 结合 NCA 和标签向量确定故障信息算法流程

Table 3 Algorithm of identifying the information of faults with NCA and signature vectors

算法 2: IFI_NCA
输入: ΔZ
输出: 故障信息

```
IFI_NCA( ΔZ )
{
    P ← Identify_Fault_Information( ΔZ );    //调用算法 1
      if( |P| == 1 ) return  P;      // |P| == 1, 表示 P 中只有一个元素
      else NCA( P, ΔZ );
}
NCA( P, ΔZ )
{
   for all  φ(k,m) ∈ P  do
      for  i from 1 to  2^12  do
         S_i ← generate an internal state randomly;
         S_i' ← flip the state of bits locating between the (k+1)_th and (k+m)_th cells in S_i
         Z_i ← let  S_i  be the initial state of PRGA phase, and generate 30 keystream bits;
         Z_i' ← let  S_i'  be the initial state of PRGA phase, and generate 30 keystream bits;
         ΔZ_i = Z_i ⊕ Z_i';
      end for
      (k,m) ← generate a table containing different ΔZ_i with corresponding frequency
   end for
   H ← the first 30 bits of  ΔZ
   find the table in which the corresponding frequency of  H  is highest
}
```

3.3.2 利用故障序列生成方程组

利用与上一小节类似的方法, 通过故障密钥流我们可以得到一组关于 Y_0 和 X_0 的方程. 假设 0 时刻, 在 k 位置上引入了一个故障, 引起了 m 个连续比特的翻转. 于是在引入故障之后的初态变为

$$S_{0,\Delta\phi_{(k,m)}}=(s_0,\cdots,s_{k-1}\oplus 1,s_k\oplus 1,\cdots,s_{k+m-1}\oplus 1,\cdots,s_{159})$$

利用 l 比特的故障密钥流可以得到 $3l$ 个关于 $S_{0,\Delta\phi_{(k,m)}}$ 的方程.

在保持 Key 和 IV 不变的前提下重置加密机, 反复在 PRGA 阶段的 0 时刻引入若干故障可以得到一系列的方程. 例如, 引入 v 个故障, 最终可以得到一组含有 $3(v+1)l$ 个方程, $160+2(v+1)l$ 个变元的方程组.

3.4 解非线性方程组

易见, 由上节方法获得的方程组是非线性多项式方程组. 目前, 常用的解非线性方程组的方法有 Gröbner 基方法, 如 F4 和 F5 算法, 以及 SAT 求解器. 本文采用 SAT 求解器来求解非线性方程, 并使用由 M. Soos 开发的 CryptMiniSat2.9.6. 首先, 将方程组转化为一个 SAT 实例, 即将方程组的代数正规型转化为一个布尔公式的合取范式 CNF, 国际上有标准的 SAT 实例格式, 称为 DIMAC 格式, 另外 CryptMiniSa2.9.6 在 DIMAC 格式基础上增加了异或输入格式; 其次, 将转化后的 SAT 实例输入 SAT 求解

器, 若 SAT 求解器返回 FALSE, 即实例“不可满足”①, 则说明方程组无解; 若 SAT 求解器返回 TRUE, 即实例“可满足”, 则说明方程组有解, 同时, SAT 求解器还会返回一组解. 需要注意的是, 当方程组存在多个解时, SAT 求解器返回的解不一定是所需要的解(比如, 我们需要的 Grain-v1 的内部状态).

为了利用 SAT 求解器求出所需的初态值, 在原先方程组的基础上每次固定一个变元的取值为 0 或 1 (或者说是猜测一个变元的取值), 然后求解新的方程组. 如果修改后的方程组无解, 那么说明这一变元的值猜测错误. 否则, 修改后的方程得到的解仍然是原来方程组的解. 重复这个过程多次, 如果得到的解数量比较少, 那么可以认为这些解给出了原方程组的全部解, 只需要逐个验证查找 PRGA 阶段初态.

4 实验结果

本节给出了上述攻击详细的实验结果. 在攻击过程中, 以下三个问题值得关注:

(1) 算法 2 对于唯一确定故障信息的准确性如何;

(2) 恢复初态需要引入多少个故障. 所需故障的数量说明了在一次攻击中需要重置加密装置的次数.

(3) 攻击所需要的密钥流长度.

本文中实验环境为 2.83GHz CPU, 4G RAM 的 PC 机, 采用的 SAT 求解器为 CryptMiniSat2.9.6.

4.1 确定故障信息的准确性

为了测试算法 2 的成功概率, 对每一个 $\phi_{(k,m)} \in \Gamma$, 生成对应的差分序列, 然后利用算法 2 确定故障的信息. 实验结果表明, 在全部 1252 种可能的故障中, 利用算法 2, 其中只有 32 种情况不能正确确定故障的信息, 即利用算法 2 可以以接近 97.5%的正确概率确定一个故障信息. 表 4 给出算法 1 和 2 成功概率的对比.

表 4 算法 1 和算法 2 成功概率比较

Table 4 The success rates of Algorithm 1 and Algorithm 2

算法	所有可能的故障总数	唯一确定故障信息个数	成功概率
算法 1	1252	301	24%
算法 2	1252	1220	97.5%

4.2 恢复中间状态所需要的故障数

由于密钥流长度会影响单个故障生成方程的数量. 因此, 我们首先研究了在不同密钥流长度下, 攻击所需要的故障的数量. 在实验中, 利用不同密钥流长度以及相应的故障数生成关于 PRGA 阶段初态的方程组. 在该方程组的基础上逐次猜测 s_0, s_1, s_2, s_3, s_4 中一个变元的取值为 0 或 1, 统计方程组的解的数量. 若方程组只有一个解, 我们认为可以利用 SAT 求解器对该组方程进行求解来恢复中间状态. 具体结果见表 5.

表 5 不同密钥流长度下攻击所需故障数量

Table 5 The number of fault under different length of keystream used in our attacks

故障个数	m 范围	密钥流长度	解平均个数	解中包含 PRGA 阶段初态的比例	求解方程平均时间(秒)
10	[1,8]	100	1	100%	2293.951
8	[1,8]	160	1	100%	2741.233

由表 5 中数据可以看出, 当密钥流长度为 100 比特(错误密钥流和正确密钥流各 50 比特)时, 利用 10 个故障生成的方程组可以认为有唯一解, 即可以利用 SAT 求解器恢复出 PRGA 阶段初态. 而当密钥流长度增加到 160 时, 只需要引入 8 个故障便可以利用 SAT 求解器恢复出 PRGA 阶段初态.

①布尔公式的可满足性是指存在一组变量的赋值使得布尔公式取值为 TRUE.

单个故障翻转比特的数量对于方程的数量也有一定的影响. 例如, 当一个故障只引起一个中间状态比特翻转, 并且该比特又出现在输出函数的线性抽头时, 分别利用正常密钥序列与错误密钥序列得到的两个方程是相同的, 降低了有效方程的数量. 当故障数为 10 时, 我们分别测试了在故障翻转比特数 $m\in[1,5]$ 和 $m\in[6,8]$ 两种情况下, 攻击所需要的密钥流长度. 具体结果见表 6.

表 6 不同故障翻转比特数下攻击所需密钥流长度
Table 6 The length of keystream bits respect to m

故障个数	m 范围	密钥流长度	解平均个数	解中包含 PRGA 阶段初态的比例	求解方程平均时间(秒)
10	[1,5]	120	1	100%	927.8781
10	[6,8]	100	1	100%	285.4390

由数据可得, 当引入的故障数都是 10 个时, 如果故障翻转的比特数 $m\in[6,8]$, 攻击者只需要利用长为 100 比特的密钥流便可以利用 SAT 求解器恢复出算法 PRGA 阶段的初态; 如果故障的翻转比特数 $m\in[1,5]$, 攻击者则需要利用长为 120 比特的密钥流来恢复出算法 PRGA 阶段的初态.

最后, 与已有攻击的比较见表 7. 特别地, 由表 7 可以看到, 在单个故障引发的比特翻转数量和所需密钥流比特数量方面, 本文攻击方法都优于文献[13].

表 7 与已有攻击的比较
Table 7 Comparions with previous attacks

攻击方法	单个故障引发的比特翻转数量	故障引入位置	攻击对象	确定故障信息成功概率	所需故障数量	所需密钥流比特数量
文献[8]	1 比特	LFSR	Grain-128	1	24	--
文献[9]	1 比特	NFSR	Grain-128	接近 1	256	--
文献[12]	1 比特②	LFSR 或 NFSR	Grain-v1	0.94③	10	160
文献[13]	最多 5 比特	LFSR 或 NFSR	Grain-128	0.93	5	160
本文	最多 8 比特	LFSR 或 NFSR, 或横跨 LFSR 和 NFSR	Grain-v1	0.97	8	160

5 总结与展望

本文针对 Grain-v1 算法, 在单个故障能够引发至多连续 8 个中间状态比特翻转的假设下, 给出了一个有效的差分故障攻击方法. 在单个故障引发比特翻转数量、所需故障数量以及所需密钥流长度方面均优于已有攻击方法. 另外, 本文攻击方法也可以用于处理单个故障引发翻转比特数量大于 8 的情形. 在下一步工作中, 我们希望能够进一步减弱攻击的假设条件, 并降低攻击所需要的故障数量.

References

[1] 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.

[2] HELL M, JOHANSSON T, MAXIMOV A, et al. A stream cipher proposal: Grain-128[C]. In: International Symposium on Information Theory—ISIT 2006. IEEE, 2006:1614–1618.

②虽然文献[12]中假设单个故障引发小于等于 3 比特翻转, 但是攻击过程中只能区分出差分序列是否由多比特故障引发, 若是多比特情形, 则确定故障的算法将输出 null, 即攻击中只利用引发 1 比特翻转的故障.

③此处的成功概率是指文献[12]中算法正确区分一个故障是否引发多个比特状态翻转的概率.

[3] AGREN M, HELL M, JOHANSSON T, et al. Grain-128a: a new version of Grain-128 with optional authentication[J]. International Journal of Wireless and Mobile Computing, 2011, 5(1): 48–59.
[4] LEE Y, JEONG K, SUNG J, et al. Related-key chosen IV attacks on Grain-v1 and Grain-128[C]. In: Information Security and Privacy. Springer Berlin Heidelberg, 2008: 321–335.
[5] RAHIMI M, BARMSHORY M, MANSOURI M H, et al. Dynamic Cube Attack on Grain-v1[J]. IACR Cryptology ePrint Archive, 2013, 2013: 268.
[6] KNELLWOLF S, MEIER W, NAYA-PLASENCIA M. Conditional differential cryptanalysis of NLFSR-based cryptosystems[C]. In: Advances in Cryptology—ASIACRYPT 2010. Springer Berlin Heidelberg, 2010: 130–145.
[7] ZHANG B, LI Z, FENG D, et al. Near collision attack on the grain v1 stream cipher[C]. In: Fast Software Encryption. Springer Berlin Heidelberg, 2014: 518–538.
[8] BERZATI A, CANOVAS C, CASTAGNOS G, et al. Fault analysis of GRAIN-128[C]. In: IEEE International Workshop on Hardware-Oriented Security and Trust—HOST 2009. IEEE, 2009: 7–14.
[9] KARMAKAR S, CHOWDHURY D R. Fault analysis of Grain-128 by targeting NFSR[C]. In: Progress in Cryptology—AFRICACRYPT 2011. Springer Berlin Heidelberg, 2011: 298–315.
[10] BANIK S, MAITRA S, SARKAR S. A differential fault attack on the grain family of stream ciphers[C]. In: Cryptographic Hardware and Embedded Systems—CHES 2012. Springer Berlin Heidelberg, 2012: 122–139.
[11] BANIK S, MAITRA S, SARKAR S. A differential fault attack on the grain family under reasonable assumptions[C]. In: Progress in Cryptology—INDOCRYPT 2012. Springer Berlin Heidelberg, 2012: 191–208.
[12] SARKAR S, BANIK S, MAITRA S. Differential fault attack against grain family with very few faults and minimal assumptions[J]. IACR Cryptology ePrint Archive 2013, 2013: 494.
[13] DEY P, CHAKRABORTY A, ADHIKARI A, et al. Multi-Bit Differential Fault Analysis of Grain-128 with Very Weak Assumptions[J]. IACR Cryptology ePrint Archive, 2014, 2014: 654.
[14] BONEH D, DEMILLO R A, LIPTON R J. On the importance of checking cryptographic protocols for faults[C]. In: Advances in Cryptology—EUROCRYPT 1997. Springer Berlin Heidelberg, 1997: 37–51.
[15] BIHAM E, SHAMIR A. Differential fault analysis of secret key cryptosystems[C]. In: Advances in Cryptology—CRYPTO 1997. Springer Berlin Heidelberg, 1997: 513–525.
[16] HOCH J J, SHAMIR A. Fault analysis of stream ciphers[C]. In: Cryptographic Hardware and Embedded Systems—CHES 2004. Springer Berlin Heidelberg, 2004: 240–253.
[17] BAR-EL H, CHOUKRI H, NACCACHE D, et al. The sorcerer's apprentice guide to fault attacks[J]. Proceedings of the IEEE, 2006, 94(2): 370–382.
[18] SKOROBOGATOV S P, ANDERSON R J. Optical fault induction attacks[C]. In: Cryptographic Hardware and Embedded Systems—CHES 2002. Springer Berlin Heidelberg, 2003: 2–12.

作者信息

叶晨东(1992–), 浙江兰溪人, 硕士. 主要研究领域为序列密码设计与分析.
E-mail: ye_chendong@126.com

田甜(1981–), 湖北武汉人, 博士, 副教授. 主要研究领域为序列密码设计与分析.
E-mail: tiantian_d@126.com