doi:10.11959/j.issn.1000-436x.2017274

轻量级分组密码算法 TWINE 差分故障攻击的改进

高杨，王永娟，王磊，王涛
（信息工程大学，河南 洛阳 471003）

摘 要：针对轻量级分组密码 TWINE 的半字节分组差分扩散规律展开研究，提出一种新的差分故障攻击的方法，并基于 S 盒差分分布统计规律性计算出恢复轮密钥的概率下界，由此给出完整恢复种子密钥的故障注入次数期望。理论证明和实验结果同时表明，算法第 33、34、35 轮平均注入 9 次故障即可完全恢复种子密钥。最后提出故障注入位置的改进，提升了实际攻击的可行性。

关键词：轻量级分组密码；TWINE 算法；差分故障攻击；概率模型
中图分类号：TP309.2 **文献标识码**：A

Improvement Differential fault attack on TWINE

GAO Yang, WANG Yong-juan, WANG Lei, WANG Tao
(Information Engineering University, Luoyang 471003, China)

Abstract: A new method of differential fault attack was proposed, which was based on the nibble-group differential diffusion property of the lightweight block cipher TWINE. On the basis of the statistical regularity of the S-box differential distribution, the lower bound of the probability of recovering round key was calculated. Then expectation of number of fault injections when restoring seed key can be estimated. Theoretical proof and experimental results both show that an average of nine times of fault injections in 33, 34 and 35 rounds bring about the seed key recovered completely. Finally, the improvement of the fault injection location was proposed, which enhances the feasibility of the genuine attack.

Key words: lightweight block cipher, TWINE algorithm, differential fault attack, probability model

1 引言

随着网络技术的进步，人们在网上通信的需求越来越大，密码技术也得到极大发展。另一方面，受限于实际生活中的加密环境与运算资源，为了达到加密算法效率与安全的双重标准，轻量级密码算法近年来得到了密码学领域研究人员的广泛关注，如 LBlock[1]、Piccolo[2]、LED[3]、MIBS[4]、HIGHT[5]、PRESENT[6]等。

TWINE 算法最早是由 Suzaki 等[7]在 SAC 2012 上提出的轻量级分组密码，分组长为 64 bit，支持 80 bit 和 128 bit 2 种密钥长度。相较其他轻量级分组密码算法，TWINE 算法采用了广义 Feistel 结构，同时应用分组随机置换，使其软件实现和硬件实现都达到了较高的效率。从运算速度和硬件规模上来看，TWINE 算法都非常适用于智能卡等资源受限的物联网加密环境。

由于 TWINE 算法的广泛应用前景，自提出以来，有许多学者用传统方法对该算法进行了分析。首先，TWINE 的设计者 Suzaki 等[7]对该算法进行了不可能差分分析和饱和攻击，虽能完整破解密钥，但时空复杂度较高。之后，Karakoç 等[8]提出了对 TWINE 算法的 Biclique 攻击；2013 年，Boztaş[9]利用算法加密过程中密钥扩散缓慢的特性提出了缩减轮数的多维中间相遇攻击。

另一方面，故障攻击早在 1996 年就由 Boneh 等[10]提出，用于攻破基于 CRT 方式实现的 RSA 签名算法。1997 年，经过 Biham 等[11]的改进，著名的差分故障

收稿日期：2017-10-03
基金项目：国家博士后科学基金面上基金资助项目（No.2014M552603）
Foundation Item: China Postdoctoral Science Foundation (No.2014M552603)

分析方法应运而生。他们凭借这种新方法成功攻击了分组密码 DES 算法。之后，人们在此基础上不断提出新的差分故障攻击方法，成功攻破了 LED[12]、MIBS[13]、Camellia[14]、Trivium[15]等密码算法。

本文对 TWINE 算法的安全性进行了分析和研究，主要的成果和创新点如下。

1) 对文献[16]的差分故障攻击方法进行了改进。共需在第 33、34、35 轮输入中注入 21 个故障位点，最少 6 次故障注入可完全恢复出 80 bit 密钥，恢复出初始密钥的故障注入次数期望降低至 8.94，较原文献效率得到极大提升。

2) 在改进方法的基础上，提出新的故障注入思路，将 1)中故障注入位点后移一轮，只需在第 34、35、36 轮输入中导入故障，提升实际攻击的可行性。

2　TWINE 算法简介

2.1　符号说明

x_i^r：第 r 轮位置为 i 的 4 bit 输入值；

K：种子密钥；

k_i：位置为 i 的 4 bit 密钥值；

W_k^r：第 r 轮全部 80 bit 轮密钥；

R_k^r：参与轮函数运算的第 r 轮 32 bit 密钥；

$R_{k_i}^r$：第 r 轮位置为 i 的 4 bit 密钥值；

C_i：位置为 i 的 4 bit 正确密文；

C_i^*：位置为 i 的 4 bit 错误密文；

ΔC_i：位置为 i 的正确密文与错误密文的差分；

f_i：位置为 i 的 4 bit 故障值；

$\#S$：集合 S 包含的元素个数；

p_l^r：$k_i = 0\text{x}1111$ 时，注入 l 次故障后恢复出第 r 轮密钥的概率。

2.2　TWINE 算法介绍

TWINE 算法有 2 个版本，分别为 TWINE-80 和 TWINE-128，分别代表 80 bit 和 128 bit 初始密钥长度。2 个版本的分组长度均为 64 bit，迭代轮数为 36 轮。该算法采用半字节基础处理单元和 Type-II 型广义 Feistel 结构[17]。

首先将要加密的消息分为 16 组，每组包含半字节（4 bit）数据。轮函数包含一层非线性变换和一层 16 组置换（最后一轮无置换），如图 1 所示。非线性变换为 8 个相同的 4 进 4 出 S 盒，置换则基于半字节的位置置乱。

3　TWINE 算法结构性质

3.1　TWINE 算法差分扩散规律

TWINE 算法每轮的置换基于 4 bit 的置乱，通过连续 2 轮的输入分析其差分扩散规律。假设第 r 轮的输入为 $x_0^r, x_1^r, x_2^r, \cdots, x_{15}^r$，经分析发现，在 x_{2k}^r 处引入的差分将扩散到下一轮互不相同的 3 个位置，分别为 x_{2k}^{r+1}、x_{2k+1}^{r+1} 和 $x_{\pi(2k)}^{r+1}$；在 x_{2k+1}^r 处引入的差分将扩散到下一轮互不相同的 2 个位置，分别为 x_{2k}^{r+1} 和 x_{2k+1}^{r+1}，如图 2 所示。

3.2　TWINE 算法 S 盒差分分布情况

在 TWINE 算法中，轮函数的非线性变换与密钥扩展算法的复杂性都基于同一个 S 盒。因此，研究该 S 盒的差分分布特性是很有必要的。给定

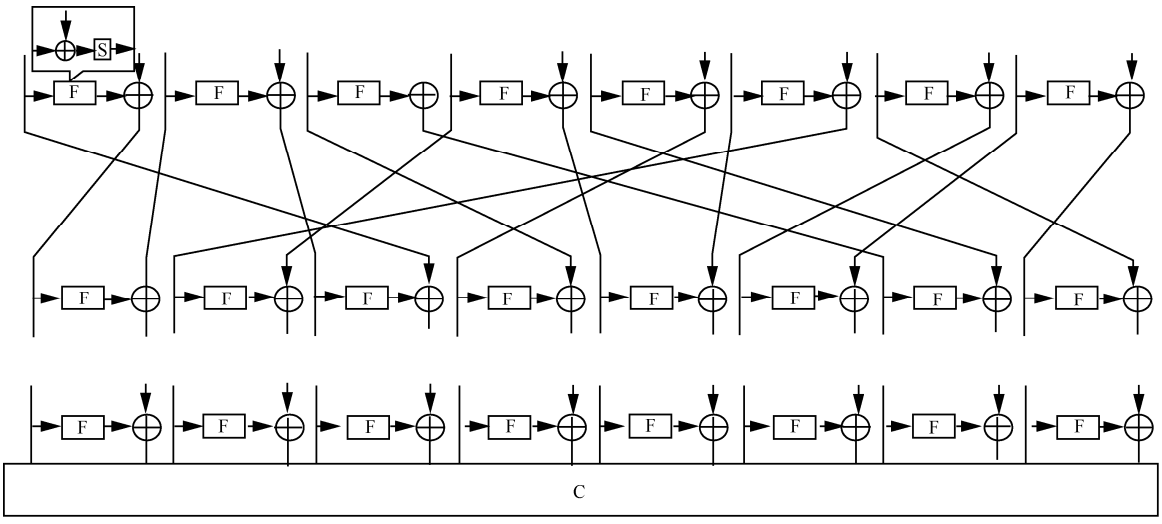


图 1　TWINE 算法结构

$\alpha \in F_2^4, \beta \in F_2^4, m \in F_2^4$，满足 $S(m \oplus a) \oplus S(m) = \beta$，则称 α 为 S 盒输入差分，β 为 S 盒的输出差分。TWINE 算法 S 盒在输入差分一定情况下输出差分与输入值的对应关系如表 1 所示。

表 1 输入差分固定情况下输出差分与输入值的对应关系

α		对应关系						
1	β	2	5	9	A	B	C	F
	m	E,F	2,3	4,5	A,B	8,9	0167	C,D
2	β	3	4	5	7	A	B	E
	m	0,2	9,B	8,A	C,E	13FD	4,6	5,7
3	β	2	5	6	7	8	E	F
	m	5,6	C,F	0,3	4,7	D,E	9,A	128B
4	β	3	6	9	B	D	E	F
	m	B,F	2,6	8,C	15AE	9,D	0,4	3,7
5	β	1	2	3	6	7	9	A
	m	B,E	149C	3,6	8,D	0,5	A,F	2,7
6	β	1	5	7	9	C	D	E
	m	3,5	0167	9,F	B,D	A,C	2,4	8,E
7	β	3	4	5	6	8	9	C
	m	A,D	2,5	9,E	B,C	3,4	0167	8,F
8	β	1	2	3	4	5	D	F
	m	7,F	2,A	149C	0,8	5,D	3,B	6,E
9	β	3	7	8	A	C	D	F
	m	7,E	3,A	128B	5,C	4,D	6,F	0,9
A	β	1	4	7	8	9	B	F
	m	0,A	4,E	128B	6,C	3,9	7,D	5,F
B	β	2	4	6	7	B	C	D
	m	3,8	7,C	4,F	6,D	0,B	2,9	15AE
C	β	2	4	8	9	A	D	E
	m	7,B	6,A	5,9	2,E	4,8	0,C	13FD
D	β	1	2	3	8	B	C	E
	m	149C	0,D	5,8	7,A	2,F	3,E	6,B
E	β	1	4	6	A	C	E	F
	m	6,8	13DF	7,9	0,E	5,B	2,C	4,A
F	β	1	5	6	8	A	B	D
	m	2,D	4,B	15AE	0,F	6,9	3,C	7,8

观察表 1，容易归纳出 TWINE 算法 S 盒的差分分布统计特性。

1) 固定输入值 m，共有 15 种可能的“输入—输出”差分对(D_{in}, D_{out})，且 D_{in} 与 D_{out} 分别跑遍 1 到 F 这 15 个数。

2) $m \neq 1$ 时，其对应的 15 组(D_{in}, D_{out})中有 3 组对应的可能输入值记为集合 $Y_i (i = 1,2,3)$，$\#Y_i = 4$；其余 12 组对应的可能输入值记为集合 $Z_i (i = 1,\cdots,12)$，$\#Z_i = 2$。

3) $Y_1 = Y_2 = Y_3$；$Z_i \cap Z_j = \{a\}$；$Y_i \cap Z_j = \{a\}$，$(i = 1,2,3; j = 1,\cdots,12)$。

4) $m = 1$ 时，其对应的 15 组(D_{in}, D_{out})可能输入值记为集合 P_i(i=1,2,⋯,15)，$\#P_i = 4$。

4 TWINE 算法的差分故障攻击

TWINE 算法的加密消息分组与 S 盒数据处理单元为半字节，结合该特点本节的差分故障攻击方法均采用基于半字节的随机故障注入模型。

在此给出攻击条件以及具体假设。

1) 攻击者可以完全掌握密码设备，可以在加密过程中任意时刻任意位置注入半字节的随机故障，但故障具体值未知。

2) 攻击者可以反复多次在同一位置重复导入随机故障。

3) 攻击者可以反复重启密码设备，加密相同的明文和初始密钥。

4.1 改进的攻击方法

4.1.1 基本攻击模型

文献[16]利用了 TWINE 算法的差分扩散规律，在第 35 轮下标为偶数的输入位置导入故障，得到了输出密文的差分扩散规律，进而恢复出最后一轮的轮密钥。

以在 x_0^{35} 处导入故障为例，首先由 TWINE 算法的轮函数得到

$$C_0 = x_0^{36} = S\left(x_0^{35} \oplus R_{k_0}^{35}\right) \oplus x_1^{35}$$

$$C_1 = S\left(C_0 \oplus R_{k_0}^{36}\right) \oplus x_2^{35}$$

$$C_5 = S\left(x_3^{35} \oplus R_{k_4}^{36}\right) \oplus x_0^{35}$$

因此，如 3.1 节分析，在 x_0^{35} 处导入故障将在 C_0、C_1、C_5 处产生错误密文。对于 C_1，由正确密文得 $C_1 = S\left(C_0 \oplus R_{k_0}^{36}\right) \oplus x_2^{35}$，由错误密文得 $C_1^* = S\left(C_0^* \oplus R_{k_0}^{36}\right) \oplus x_2^{35}$，联立两式可得差分方程

$$\Delta C_1 = S\left(C_0 \oplus R_{k_0}^{36}\right) \oplus S\left(C_0^* \oplus R_{k_0}^{36}\right) \tag{1}$$

式(1)中仅 $R_{k_0}^{36}$ 未知，综上所述，在 x_0^{35} 处导入故障可以解出 $R_{k_0}^{36}$ 的值。

文献[16]给出，若式(1)有解，则 84.9%的概率有 2 个解，14.2%的概率有 4 个解。故可对 x_0^{35} 反复导入多次故障，求各次 $R_{k_0}^{36}$ 解空间的交集可以较高概率求出 $R_{k_0}^{36}$ 的具体值。

同理可以通过差分扩散规律得知在第 35 轮其他位置处导入故障后产生错误密文的位置。经过分析计算可以得到其余 7 个差分方程

$$\Delta C_{i+1} = S\left(C_i \oplus R_{k_i}^{36}\right) \oplus S\left(C_i^* \oplus R_{k_i}^{36}\right)$$

$$(i = 2,4,6,8,10,12,14)$$

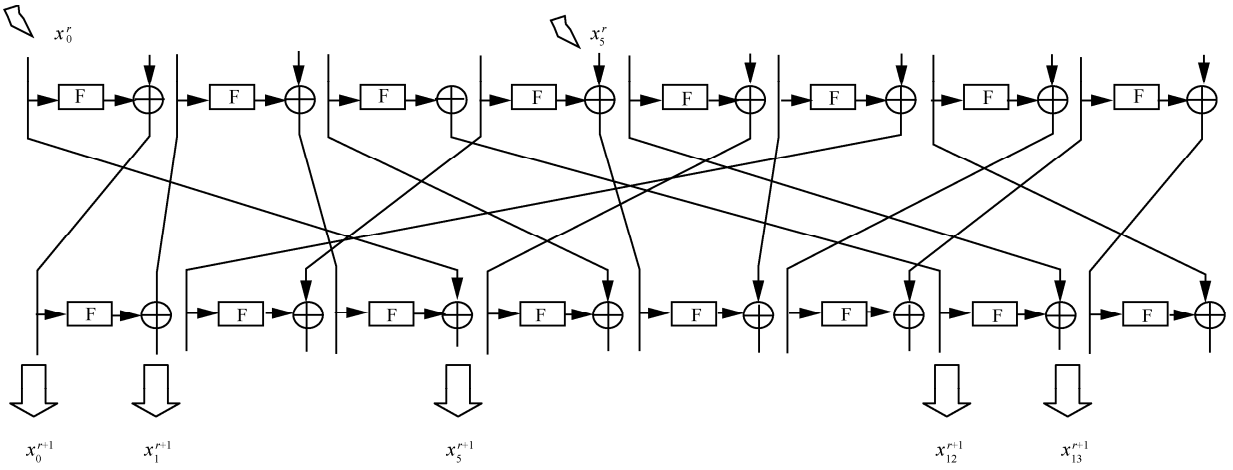


图 2　TWINE 算法差分扩散规律

为避免注入故障对错误密文的重复影响，进而导致差分方程不可解，文献[16]将第 35 轮 8 个位置分为 4 组 $T_1=\{x_0^{35},x_2^{35}\}$，$T_2=\{x_4^{35},x_8^{35}\}$，$T_3=\{x_6^{35},x_{10}^{35}\}$，$T_4=\{x_{12}^{35},x_{14}^{35}\}$，同时从每组任取一个位置导入随机故障，通过差分方程求解得到 R_k^{36} 4 个半字节 $(R_{k_0}^{36},R_{k_{12}}^{36},R_{k_8}^{36},R_{k_{10}}^{36})$的值。再在每组剩余 4 个位置中同时导入故障，以求得 R_k^{36} 其余 4 个半字节 $(R_{k_4}^{36},R_{k_6}^{36},R_{k_2}^{36},R_{k_{10}}^{36})$的值。

4.1.2　注入故障方法的改进

上述方法选取的是能将错误扩散至三个半字节的位置，虽然扩散效果好，但是因为错误密文在差分方程中的重复出现而对故障动作有较强的限制性。文献[16]中，需要在第 35 轮输入的相同位置导入 2 次故障求取 R_k^{36} 中 4 个半字节的值。因此，至少要导入 4 次故障才能完整恢复 R_k^{36}，相对比较繁琐。为了简化攻击步骤，本文优化了导入故障的方法。

由图 2 知，任意 2 个形如 x_{2m+1}^{35} 位置对应的扩散输出半字节均不相交，且构成了输出密文 C 完整的 16 个半字节分组。因此，本文选择在所有 x_{2m+1}^{35} 位置导入故障，最少通过 2 次故障注入即可求取 R_k^{36}。用 R_k^{36} 解密密文 C，可以得到第 35 轮中间输出状态，采取上述方法求得 R_k^{35} 和 R_k^{34}；另一方面，可以用 R_k^{36} 表示出 W_k^{36}，进而根据密钥扩展算法表示出 W_k^{35} 和 W_k^{34}，最终得到的 W_k^{34} 可完全用 $R_{k_i}^r\ (r=34,35,36)$ 表示，再通过密钥扩展方案即可反推出种子密钥 K。

4.1.3　攻击流程

1) 故障注入

① 任意选择明文 P，在初始密钥 K(80 bit) 的作用下加密，并正确获取正确密文 $C=C_0\|C_1\|C_2\|C_3\|C_4\|\cdots\|C_{14}\|C_{15}$。

② 使用相同明文及密钥进行加密运算，在运算至第 35 轮时，分别在 $x_{2i+1}^{35}(i=0,1,\cdots,7)$ 处同时导入半字节随机故障，并记录错误密文 $C^*=C_0^*\|C_1^*\|C_2^*\|C_3^*\|C_4^*\|\cdots\|C_{14}^*\|C_{15}^*$。

2) 列出 S 盒差分方程

$$\Delta C_{i+1}=S(C_i\oplus R_{k_i}^{36})\oplus S(C_i^*\oplus R_{k_i}^{36}),\quad (i=0,2,4,6,8,10,12,14)$$

3) 筛选密钥值

① 对步骤 2)中每个差分方程，穷举 4 bit 密钥值，对满足等式的密钥值列入候选值集合 $N_i\ (i=0,2,4,6,8,10,12,14)$。

② 再次进行步骤 1)中的步骤②，列出方程并筛选密钥列入集合 $Q_i\ (i=0,2,4,6,8,10,12,14)$，取 Q_i 与 N_i 的交集即为 $R_{k_0}^{36},R_{k_2}^{36},R_{k_4}^{36},R_{k_6}^{36},R_{k_8}^{36},R_{k_{10}}^{36},R_{k_{12}}^{36},R_{k_{14}}^{36}$ 的正确值。

4) 恢复 R_k^{35} 和部分 R_k^{34}

① 用已经得到的 R_k^{36} 解密密文，获得第 35 轮正确中间状态输出 x^{35}。在算法运行至第 34 轮时，分别在 $x_{2i+1}^{34}\ (i=0,1,\cdots,7)$ 处同时导入半字节随机故障，重复上述步骤，恢复 32 bit R_k^{35}。

② 用 R_k^{35} 解密 x^{35}，得到第 34 轮正确中间状态 x^{34}。再次运行算法，在 $x_3^{33},x_9^{33},x_7^{33},x_5^{33},x_{15}^{33}$ 处注入半字节随机故障得到错误中间状态输出 x^{34*}，根据差分方程恢复 $R_{k_4}^{34}$、$R_{k_6}^{34}$、$R_{k_8}^{34}$、$R_{k_{12}}^{34}$、$R_{k_{14}}^{34}$ 的值。

5) 根据密钥扩展方案反推出种子密钥 K

由 4.1.2 节分析知，W_k^{34} 可完全用 $R_{k_i}^r\ (r=34,35,36)$

表示，即

$$\left(R_{k_{12}}^{36}\oplus\left(0\|CON_L^{35}\right),R_{k_{14}}^{35}\oplus S\left(R_{k_{12}}^{36}\oplus\left(0\|CON_L^{35}\right)\right),R_{k_8}^{36},R_{k_{10}}^{36}\right)$$

$$\left(R_{k_4}^{34},R_{k_0}^{35},R_{k_6}^{34},R_{k_2}^{35}\oplus\left(0\|CON_H^{34}\right)\right)$$

$$\left(R_{k_4}^{35},R_{k_0}^{36},R_{k_6}^{35},R_{k_2}^{36}\oplus\left(0\|CON_H^{35}\right)\right)$$

$$\left(R_{k_4}^{36},R_{k_8}^{34},R_{k_6}^{36},R_{k_{12}}^{34}\right)$$

$$\left(R_{k_{14}}^{34},R_{k_8}^{35},R_{k_{10}}^{35},R_{k_{12}}^{35}\oplus\left(0\|CON_L^{34}\right)\right)$$

而 $R_{k_i}^r$ 均已知，由密钥方案伪代码逆推，最终可得到 80 bit 初始密钥 K。

4.1.4 复杂度分析

依照 4.1.3 节的攻击具体步骤，最理想的情况下只需注入 2+2+2=6 次故障即可完全恢复初始密钥 K。但由于 TWINE 算法 S 盒差分分布的不均匀性，针对某一差分方程，导入 2 次故障后 Q_i 与 N_i 交集中的元素可能不唯一，此时无法确定对应半字节轮密钥值。

为此，本文将 4.1.3 节出现的差分方程简化为 $S[m]\oplus S[m\oplus f]=f'$，其中，$m$ 为 S 盒的未知输入，f 为导入的随机故障值，f' 为输出差分。由 3.2 节统计特性 3)、4)可知，当且仅当 2 次注入故障后得到差分对$(D_{\text{in}},\ D_{\text{out}})$所对应可能输入值集合完全相同时，差分方程解的个数为 4；其余情况下，均能确定 S 盒唯一输入值 m。当输入值 $m=1$时，2 次故障注入得到差分对$(D_{\text{in}},\ D_{\text{out}})$与 $(D'_{\text{in}},D'_{\text{out}})$所对应可能输入值集合完全相同的概率为 $\frac{3}{15}=0.2$；当输入值 $m\neq1$时，两集合相等的概率降低至 $\left(\frac{3}{15}\right)^2=0.04$。即当 $m=1$时，在注入故障次数相同情况下差分方程有解的概率更低。因此，仅考虑输入值 $m=1$的特殊情况，可以得到恢复轮密钥 R_k 的一个下界。

定理 1 在 4.1.2 节的故障注入模型下，导入 l 次故障后恢复出 R_k^r 的概率下界为

$$p_l^r=1-\sum_{i=1}^{8}\frac{(-1)^{i+1}\mathrm{C}_8^i}{5^{li}}$$

证明 针对任一差分方程，只考虑 $m=1$的特殊情况。由乘法原理可知，8 个差分方程均有唯一解的概率为 $p_2^r=\left(1-\frac{3}{15}\right)^8\times100\%=16.78\%$，该概率即为 2 次导入故障恢复出 R_k^r 的概率，此时 R_k^r 的具体值为 0x11111111。由上分析知，当 R_k^r 取其他值时，被恢复出的概率应大于 16.78%。

下面考虑 $l\geqslant3$ 时的情形。对于某一差分方程，连续 3 次注入故障后得到差分对$(D_{\text{in}},D_{\text{out}})$所对应可能输入值集合完全相同时，差分方程解的个数为 4，故第 $i\left(i=1,2,\cdots,8\right)$ 个差分方程解不唯一的概率均为 $\frac{3^2\times15^{14}}{15^{16}}$，由于每个差分方程是否有唯一解是相互独立的，故第 $i,j\left(i,j=1,2,\cdots,8,i\neq j\right)$ 个差分方程解同时不唯一的概率为 $\frac{3^4\times15^{12}}{15^{16}}$，具体情况如图 3 所示。以此类推，由容斥原理可知至少一个差分方程有不唯一解的概率为

$$\sum_{i=1}^{8}\frac{\mathrm{C}_8^i\cdot3^{2i}\cdot15^{16-2i}}{15^{16}}\cdot\left(-1\right)^{i+1}=0.278\,6$$

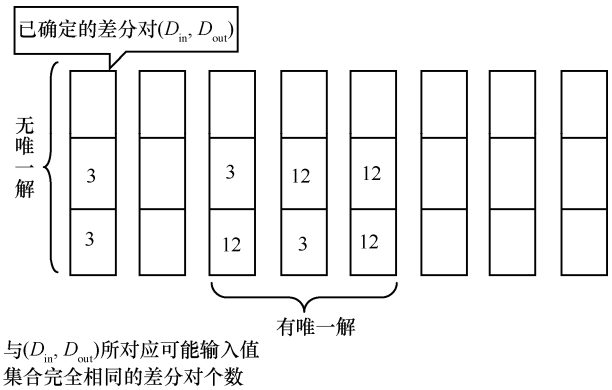


图 3 $l=3$ 时差分方程解的情形

因此，注入 3 次故障后恢复出 R_k^r 的概率至少为 $p_3^r=(1-0.278\,6)\times100\%=72.14\%$。同理归纳可得注入 l 次故障后至少一个差分方程有不唯一解的概率为

$$\sum_{i=1}^{8}\frac{\mathrm{C}_8^i\cdot3^{li}\cdot15^{8l-li}}{15^{8l}}\cdot\left(-1\right)^{i+1}=\sum_{i=1}^{8}\frac{\left(-1\right)^{i+1}\cdot\mathrm{C}_8^i}{5^{li}}$$

其对立事件，即注入 l 次故障后恢复出 R_k^r 的概率即为 $p_l^r=1-\sum_{i=1}^{8}\frac{\left(-1\right)^{i+1}\cdot\mathrm{C}_8^i}{5^{li}}$，此时，$R_k^r$ 的具体值为 0x11111111。由上分析知，当 R_k^r 取其他值时，被恢复出的概率应大于 p_l^r。

更进一步地，具体攻击步骤中只用恢复出 R_k^{34} 的 5 个半字节，注入 l 次故障恢复第 34 轮 20 bit 部分轮密钥的概率为 $1-\sum_{i=1}^{5}\frac{\left(-1\right)^{i+1}\cdot\mathrm{C}_5^i}{5^{li}}$。

为了求得恢复种子密钥 K 的概率下界，需要知道恰好导入 l 次故障恢复轮密钥 R_k^r 的概率，用 $p_l'^r$ 表示。易知 $p_2'^r = p_2^r$，当 $l \geqslant 3$ 时，由事件的互斥性可得 $p_l'^r = p_l^r - p_{l-1}^r$。由实验结果知，单轮故障注入次数 $l \geqslant 6$ 时，p_l^r 大于 98%且增幅微弱，此时恢复种子密钥 K 的概率至少为 90%，因此，在计算概率下界时，只考虑 $l < 6$ 的情形。

定理 2　在 4.1.2 节的故障注入模型下，恰好注入 $L(L = 6,7,\cdots,15)$ 次故障后恢复出种子密钥 K 的概率下界为 $p_L = \sum_{l_1+l_2+l_3=L} p_{l_1}'^{34} \cdot p_{l_2}'^{35} \cdot p_{l_3}'^{36}$，其中 $l_1, l_2, l_3 \in \{2,3,4,5\}$。

证明　根据 4.1.2 节的故障注入模型，恢复出 34,35,36 轮密钥即可反推出种子密钥 K。因此，故障注入总次数等于恢复出 $R_k^r\,(r = 34,35,36)$ 所需注入故障次数之和。利用乘法原理和加法原理，可计算出 R_k^r 为 0x11111111 时注入 L 次故障后恢复出种子密钥 K 的概率 $p_L = \sum_{l_1+l_2+l_3=L} p_{l_1}'^{34} \cdot p_{l_2}'^{35} \cdot p_{l_3}'^{36}$，即为概率下界。

p_L 的具体数值如图 4 所示。

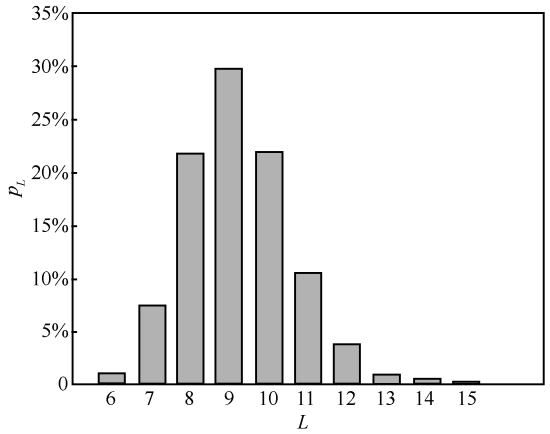


图 4　L 与 p_L 的关系

另定义至多注入 $L(L = 6,7,\cdots,15)$ 次故障后恢复出种子密钥 K 的概率下界 $\hat{p}_L = \sum_{l_1+l_2+l_3 \leqslant L} p_{l_1}'^{34} \cdot p_{l_2}'^{35} \cdot p_{l_3}'^{36}$。$\hat{p}_L$ 的具体数值如图 5 所示。

从图 4 可以看出，当 $L = 9$ 时，p_L 到达峰值；当 $L > 15$ 时，p_L 数值较小可以忽略不计。因此，可以用定理 2 中的概率下界估算出恢复种子密钥 K 所需注入故障的次数期望为 $E(L) = \sum_{L=6}^{15} L \cdot p_L \approx 8.94$。

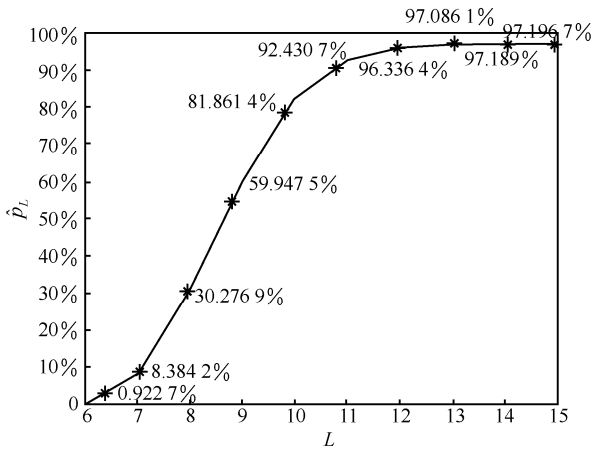


图 5　L 与 $\hat{p}_L$ 的关系

4.2　故障注入模型的进一步改进

根据 2.1 节中的 TWINE 算法差分扩散规律，在第 M 轮的输入值 x_i^M 处注入故障，其扩散效果等价于在第 M+1 轮输入值 $x_{\pi(i)}^{M+1}$ 处注入故障。运用此思路可直接将 3.1 节中故障注入位点向后推移一轮。以在第 35 轮注入故障为例，由于置换 $\pi(i)$ 将奇数全部映射成偶数，因此，在 $x_i^{35}(i = 1,3,\cdots,15)$ 处同时导入半字节随机故障等价于在 x_{i-1}^{36} 处导入故障。

5　实验仿真结果及分析

5.1　实验环境

硬件配置为一台 PC 机（CPU 为 Intel Core i5-4200M 2.5 GHz，操作系统为 64 位，内存为 4 GB），编程环境为 Microsoft Visual Studio 2012 平台下的 Visual C++语言。

5.2　实验结果

明文选取 1，2，3，4，5，6，7，8，9，A，B，C，D，E，F，0；80 bit 种子密钥随机选取。为了消除模拟注入故障过程中的人工干预，采用轻量级流密码 Trivium 生成 3 000 bit 长伪随机 01 数据流，随机截取连续 32 bit 作为故障值。模拟 150 次实验，实验结果如图 6 所示。经计算，恢复出种子密钥 K 所需注入故障均值为 9.18 次，与理论计算值较为接近。

6　结束语

本文基于 TWINE 算法的差分扩散规律，提出了一种对该算法半字节分组注入故障进行攻击的策略，给出了大幅降低故障注入次数的方法，同时

对自己提出的方法进一步改进，在保持注入故障总次数不变情况下，将故障注入位置后移一轮，增加了该攻击实现的可行性。算法复杂度方面，运用相关数学知识给出了恢复出 R_k^r 的概率下界，并估计出恢复种子密钥 K 所需注入故障的次数期望 8.94，较前人攻击结果有较大的提升，同时运用软件进行了仿真验证，实验结果与理论预期值较为接近。

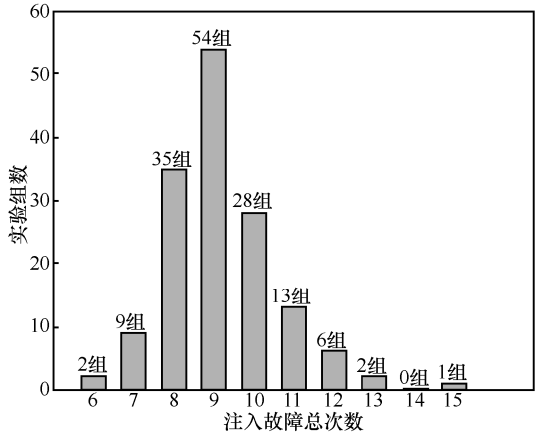


图 6　注入故障总次数对应的实验组数

本文提出的差分故障攻击方法复杂度证明为以后研究其他轻量级密码算法的故障攻击提供了思路，下一步的研究将着力于求取恢复轮密钥概率的精确值。另一方面，研究减小故障宽度的方法，考虑如何注入少量的故障恢复 R_k^r，提升故障攻击效率和可行性，同时关注 TWINE 算法如何抵抗差分故障攻击。

参考文献：

[1] WU W, ZHANG L. LBlock: a lightweight block cipher[C]//Lecture Notes in Computer Science, 7707. Berlin: Springer-Verlag, 2013: 339-354.

[2] SHIBUTANI K, ISOBE T, HIWATARI H, et al. Piccolo: an ultra-lightweight blockcipher[C]//International Conference on Cryptographic Hardware and Embedded Systems. Springer-Verlag, 2011: 342-357.

[3] GUO J, PEYRIN T, POSCHMANN A, et al. The LED block cipher[C]//International Conference on Cryptographic Hardware and Embedded Systems. Springer-Verlag, 2011:326-341.

[4] IZADI M, SADEGHIYAN B, SADEGHIAN S S, et al. MIBS: a new lightweight block cipher[J]. Lecture Notes in Computer Science, 2009, 5888:334-348.

[5] HONG D, SUNG J, HONG S, et al. HIGHT: a new block cipher suitable for low-resource device[C]//International Conference on Cryptographic Hardware and Embedded Systems. Springer-Verlag, 2006:46-59.

[6] BOGDANOV A, KNUDSEN L R, LEANDER G, et al. PRESENT: an ultra-lightweight block cipher[J]. Lecture Notes in Computer Science, 2007, 4727:450-466.

[7] KOBAYASHI E, SUZAKI T, MINEMATSU K, et al. TWINE: a lightweight block cipher for multiple platforms[C]//Conference on Selected Areas in Cryptography. 2012.

[8] KARAKOÇ F, DEMIRCI H, HARMANCI A E. Biclique cryptanalysis of LBlock and TWINE[J]. Information Processing Letters, 2013, 113(12):423-429.

[9] ÖZKAN B, KARAKOÇ F, ÇOBAN M. Multidimensional meet-in-the-middle attacks on reduced-round TWINE-128[C]// International Workshop on Lightweight Cryptography for Security and Privacy. Springer Berlin Heidelberg, 2013:55-67.

[10] BONEH D, DEMILLO R A, LIPTON R J. On the importance of checking cryptographic protocols for faults[C]//Proc of EUROCRYPT. 1997:37-51.

[11] BIHAM E, SHAMIR A. Differential fault analysis of secret key cryptosystems[C]// International Cryptology Conference. Springer, Berlin, Heidelberg, 1997:513-525.

[12] JEONG K, LEE C. Differential fault analysis on block cipher LED-64[C]// International Conference on Network-Based Information Systems. IEEE Computer Society, 2012:675-680.

[13] 赵新杰, 王韬, 王素贞, 等. MIBS 深度差分故障分析研究[J]. 通信学报, 2010, 31(12):82-89.
ZHAO X J, WANG T, WANG S Z, et al. Research on deep differential fault analysis against MIBS[J]. Journal on Communications, 2010, 31(12):82-89.

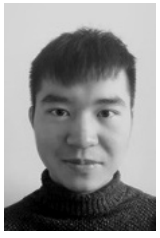
[14] 赵新杰, 王韬, 郭世泽. 一种针对 Camellia 的改进差分故障分析[J]. 计算机学报, 2011, 34(4):613-627.
ZHAO X J, WANG T, GUO S Z. An improved differential fault analysis on camellia[J]. Chinese Journal of Computers, 2011, 34(4):613-627.

[15] LI Q, GOMISAWA S, IWAMOTO M, et al. New differential fault analysis on trivium based on setup-time violations[R]. Technical Report of IEICE Isec, 2010, 110:333-339.

[16] 徐朋, 魏悦川, 潘晓中. 轻量级分组密码 TWINE 的差分故障攻击[J]. 计算机应用研究, 2015, 32(6):1796-1800.
XU P, WEI Y C, PAN X Z. Differential fault attack on TWINE[J]. Application Research of Computers, 2015, 32(6):1796-1800.

[17] SUZAKI T, MINEMATSU K. Improving the generalized Feistel[C]// Lecture Notes in Computer Science. 2010:19-39.

作者简介：



高杨（1994-），男，河南洛阳人，信息工程大学硕士生，主要研究方向为密码算法分析。

王永娟（1982-），女，河南开封人，信息工程大学副教授、硕士生导师，主要研究方向为密码算法分析。

王磊（1972-），男，河南驻马店人，信息工程大学讲师，主要研究方向为信息安全。

王涛（1995-），男，山东临沂人，信息工程大学硕士生，主要研究方向为对称密码算法的设计与分析。