

# 一种对 AES-128 的差分错误分析原理

杜育松, 王大星, 沈 静

(广州大学数学与信息科学学院, 广州 510006)

**摘 要:** 描述了一种对 AES-128 的差分错误分析原理, 给出了攻击的算法, 分析了算法成功的概率。该算法可以得到 AES-128 的中间加密结果  $M^9$ , 利用  $M^9$  和一组正确密文可以推出 AES-128 的最后一轮轮密钥, 从而恢复 AES-128 的初始密钥。软件模拟结果表明, 在物理技术达到的情况下, 如果能向  $M^9$  中反复随机地引入 140 个比特错误, 那么找到初始密钥的可能性将超过 90%。最后指出以密文分组链模式工作的 AES 可以抵抗以上提到的攻击。

**关键词:** AES; 侧信道攻击; 差分错误分析; 智能卡

## Principle of a Kind of Differential Fault Analysis on AES-128

DU Yusong, WANG Daxing, SHEN Jing

(School of Mathematics and Information Science, Guangzhou University, Guangzhou 510006)

**【Abstract】** The principle of a kind of differential fault analysis on AES-128 is described. The attacking algorithm is given. The probability of accomplishing the algorithm is analyzed. The intermediate encrypted result  $M^9$  is obtained by the algorithm, the last round key of AES-128 can be deduced from  $M^9$  and a correct cipher text, the initial key of AES-128 is then recovered. The result of software emulation shows that if one induces 140 bit faults into  $M^9$  repeatedly and randomly, the probability of recovering the initial key is more than 90%, when physical techniques are available. It is presented that AES with CBC mode can counter the attack mentioned above.

**【Key words】** AES; Side channel attack; Differential fault analysis; Smart card

1997 年, Biham 和 Shamir 提出了一种对私钥密码体制的攻击, 并命名为差分错误分析(Differential Fault Analysis)<sup>[1]</sup>。此后, 差分错误分析成为检测智能卡(Smart card)安全的考虑因素。2000 年, AES 成为 DES 的继任者, 并被越来越多地应用到智能卡中。但是, Biham 和 Shamir 提出的分析方法并不适用于 AES, 因为 AES 不是 Fistel 型密码。

2003 年, C. Giraud 提出了一种针对 AES-128 的差分错误分析原理<sup>[3]</sup>。该分析方法基于比特错误。这种错误类型首先出现在文献[1]中, 并由 Skorobogatov 和 Anderson 首先在 2002 年初步实现<sup>[5]</sup>。本文详细描述了这种基于比特错误的差分错误分析原理, 给出了具体的攻击算法, 分析了算法成功的概率, 并以软件模拟该攻击的全过程, 最后指出抵抗该攻击的对策。

### 1 高级加密标准(AES)

简单地描述 AES, 对于更多信息可以参考文献[2]。

AES 的每一数据分组为 128bits, 可以分别使用 128 bits、192 bits 和 256 bits 的不同长度的密钥加密。使用不同长度的密钥会有不同轮数的加密过程。用 128 bits 的密钥会进行 10 轮加密, 把它记为 AES-128。

每一数据分组被分成 16B, 用  $4 \times 4$  的状态矩阵表示, 每完成一轮的加密就进入到下一状态。除了最后一轮加密没有 MixColumns(MC)运算, 其余的每一轮加密包括 4 个运算: SubBytes(SB), ShiftRows(SR), MixColumns(MC)和 AddRoundKey(AR)。

每一轮加密需要一个轮密钥, 轮密钥由密钥调度算法产生。图 1 给出了 AES-128 的最后两轮加密流程。

引入比特错误

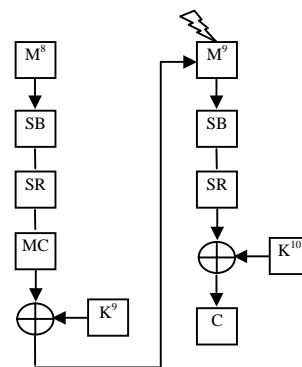


图 1 AES-128 的最后两轮加密流程

本文使用以下记号来描述 AES-128:

- (1)  $N_r$ : 加密轮数, 对于 AES-128,  $N_r=10$ ;
- (2)  $M$ : 一个明文分组(128bits);
- (3)  $M^i$ : 明文分组  $M$  在第  $i$  轮加密后的结果( $i=0,1,\dots,N_r$ );
- (4)  $M_j^i$ :  $M^i$  的第  $j$  个字节( $j=0,1,\dots,15$ );
- (5)  $K^i$ : 第  $i$  轮的轮密钥( $i=0,1,\dots,N_r$ ),  $K^0$  表示初始密钥;
- (6)  $C$ : 明文分组  $M$  对应的密文;
- (7)  $C_j$ :  $C$  的第  $j$  个字节( $j=0,1,\dots,15$ );
- (8)  $F$ : 明文分组  $M$  对应的错误密文;
- (9)  $F_j$ :  $F$  的第  $j$  个字节( $j=0,1,\dots,15$ ).

**作者简介:** 杜育松(1982 -), 男, 硕士生, 主研方向: 密码学; 王大星、沈 静, 硕士生

**收稿日期:** 2006-04-04 **E-mail:** yusongdu@21cn.com

## 2 基于比特错误的差分错误分析

比特错误是指在加密的中间结果中仅引入一比特的错误,而保证其它比特位不变。文献[3]指出,对于AES-128,如果攻击者反复地向即将进入最后一轮加密的中间加密结果 $M^9$ 中引入比特错误(如图1所示),即每次加密引入的错误使得 $M^9$ 中的某一比特改变,而保证其它比特不变。那么他有可能得到最后一轮的轮密钥。

假设错误发生在 $M_j^9$ 中,用 $M_j^9 \oplus e$ 来表示发生了错误的字节,这里 $e \in \{0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80\} = E$ 。攻击首先基于下面的事实<sup>[3]</sup>:

如果比特错误发生在 $M_j^9$ 中,那么

$$SB(M_j^9) \oplus SB(M_j^9 \oplus e) = C_{SR(j)} \oplus F_{SR(j)} \neq 0$$

具体的推导过程见文献[3]。

由上述结论可知,如果有对M的正确密文C,我们可以确定 $M^9$ 中发生比特错误的位置。假设已有对M的正确密文C以及 $F^1, F^2, \dots, F^n$  n个错误密文,且错误都发生在 $M^9$ 中的同一个字节j中,则可以由以下的算法得到 $M_j^9$ :

### 算法 1

Get\_  $M_j^9$

Input  $F^1, F^2, \dots, F^n, j$

Output  $M_j^9$  或 failure

(1) for i=1 upto n

$S_i := \{x | x \in [0, 255], e \in E, \text{且满足}$

$$SB(x) \oplus SB(x \oplus e) = C_{SR(j)} \oplus F_{SR(j)}^i\}$$

(2) if  $\left| \bigcap_{i=1}^n S_i \right| = 1$

then return  $\bigcap_{i=1}^n S_i$

else return failure

算法1能够得到 $M_j^9$ ,因为真正的 $M_j^9$ 总是满足算法1中的等式。

如果 $F^1, F^2, \dots, F^n$ 是n个不同的错误密文,检验发现,当 $n=1$ 时,算法1不可能成功。因为对于一个确定的 $F^i$ 满足 $SB(x) \oplus SB(x \oplus e) = C_{SR(j)} \oplus F_{SR(j)}^i$ 的解或没有或至少有两个。当 $n=2$ 时,算法1成功的概率约为0.90,也就是说,只要有二个发生在 $M_j^9$ 中的不同的比特错误,成功得到 $M_j^9$ 的可能性有90%。而当 $n=3$ 时,成功的概率几乎为1。

我们的检验结果与C.Giraud在文献[3]中提到的不同。文献[3]中估计的结果仅考虑最坏的情况,并得到当 $n=2$ 时的概率为0.50,当 $n=3$ 时的概率为0.97。事实上,我们的检验结果要精确一些。

注意到上述的成功概率是以假设每个比特错误都不相同为前提的。更合理的考虑是,并不能很好地保证 $F^1, F^2, \dots, F^n$ 是n个不同的错误密文,即攻击者在反复随机地向 $M_j^9$ 引入比特错误时,并不能保证每次引入的比特错误都发生在不同的比特位。基于这一事实,重新估计了算法1成功的概率。表1给出了这一结果。

表1 算法1的成功概率

错误数量 k	成功概率 $P_k$
2	$P_2=0.7875$
3	$P_3=0.9516$
4	$P_4=0.9885$
5	$P_5=0.9972$
6	$P_6=1$

这里只写出 $P_4$ 的推导过程,其余的类似。假设 $M_j^9$ 中任何一个比特发生错误的概率相等,即 $1/8$ 。如果现在有4个错误发生在 $M_j^9$ 中,可以分成4种情况,即4个错误发生在4个不同的比特上,4个错误发生在3个不同的比特上,4个错误发生在2个不同的比特上以及4个错误全发生在1个比特上。利用多项分布的性质可得

$$P_4 = \binom{8}{4} \cdot 4! \cdot 1 + \binom{8}{3} \binom{3}{1} \cdot 12 \cdot 1 + \binom{8}{2} \cdot 6 \cdot 0.90 + \binom{8}{2} \binom{2}{1} \cdot 4 \cdot 0.90 + \binom{8}{1} \cdot 0 \cdot \left(\frac{1}{8}\right)^4$$

这里0.90是同一字节恰好发生过两个比特错误且互不相同时的成功概率,相应地,1表示恰好发生过两个以上的比特错误且互不相同时的成功概率。

## 3 攻击算法

反复使用算法1可以得到 $M^9$ 中的每一个字节,这需要反复随机地向 $M^9$ 中引入多个比特错误 $F^1, F^2, \dots, F^n$ 和一个对M的正确密文C。

基于算法1,给出下面的算法2,利用它可以得到整个 $M^9$ 。在这之前,先定义函数GetErrorLocation(F),它返回对应错误密文F的比特错误位置(在哪个字节)。这一点由第2节的结论容易实现。

### 算法 2

Get\_  $M^9$

Premise:  $LastS_0 = LastS_1 = \dots = LastS_{15} = I$

Input:  $F^1, F^2, \dots, F^n, C$

Output:  $M^9$  或 failure

(1) for i=1 upto n

$L := \text{GetErrorLocation}(F^i)$

$S_L := \{x | x \in [0, 255], e \in E, \text{且满足}$

$$SB(x) \oplus SB(x \oplus e) = C_{SR(L)} \oplus F_{SR(L)}^i\}$$

$S_L := S_L \cap LastS_L$

$LastS_L := S_L$

(2) for L=0 upto 15

if  $|LastS_L| = 1$

then return  $M_L^9 := LastS_L$

else return failure

算法2中 $i$ 表示 $[0, 255]$ 上的全体整数。算法2的实现是基于使用算法1成功得到 $M^9$ 的每一个字节。

根据算法1的成功概率,下面分析算法2的成功概率。用 $x_j$ 表示发生在 $M_j^9$ 中的比特错误数量,那么有

$$\sum_{j=0}^{15} x_j = n \quad (1)$$

把在式(1)中的 $x_j$ 满足 $x_j = k$  ( $0 \leq j \leq 15, 0 \leq k \leq n$ )的个数记为 $y_k$ 且有 $y_k = 0$  ( $k=0, 1, \dots, n$ ),则 $y_k$ 可以表示向 $M^9$ 中引入n个比特错误后, $M^9$ 中恰好发生过k个比特错误的字节数。

由算法1的分析知道,只有当 $x_j = 2$  ( $j=0, 1, \dots, 15$ )时算法2才有可能成功。也就是说,当 $y_0 > 0$ 或 $y_1 > 0$ 时算法2的成功概率为零。所以只需考虑 $y_0 = y_1 = 0$ 的情况,于是有

$$\sum_{k=2}^n y_k = 16 \quad (2)$$

$$\sum_{k=2}^n k \cdot y_k = n \quad (3)$$

假设 $M^9$ 中的任何一个比特发生错误的概率都相同,即 $1/128$ ,那么由多项分布的性质可得满足式(2)和式(3)的一组解恰好为 $y_k = a_k$  ( $k=2, 3, \dots, n$ )的概率为

$$f(a_2, a_3, \dots, a_n) = \frac{\binom{16}{a_2} \binom{16-a_2}{a_3} \dots \binom{16-a_2-a_3-\dots-a_{n-1}}{a_n} \cdot n!}{\prod_{k=2}^n (k!)^{a_k} \cdot (128)^n} \quad (4)$$

假设 $M^9$ 中每个字节对于算法1能否成功是相互独立的,那么,此时算法2成功的概率为

$$f(a_2, a_3, \dots, a_n) \cdot \prod_{k=2}^n P_k^{a_k} \quad (5)$$

这里的 $P_k$ 同表1。

再由Bayes公式得到算法2总的成功概率为

$$\sum_{\sum_{k=2}^n y_k = 16} (f(y_2, y_3, \dots, y_n) \cdot \prod_{k=2}^n P_k^{y_k}) \quad (6)$$

实际上,在反复随机地向 $M^9$ 中引入比特错误后, $M^9$ 中每个字节对于算法1能否成功是相关的,所以实际的概率应略不同于式(5),算法2的成功概率也应不同于式(6)。但是,将看到,这一误差并不影响分析算法2的成功概率与引入错误数量的关系。

当引入的比特错误数 $n$ 增大时,算法2成功的概率也将增大。因为当 $n$ 增大时,同时满足式(2)和式(3)的解的数量增多,即式(6)中正的求和项增多,总概率增大。

事实上,如果把 $n$ 写成 $n=2 \times 16+r$ 的形式,那么式(1)可以改写成:

$$\sum_{j=0}^{15} x'_j = r \quad (7)$$

这里 $x'_j = x_{j-2}$  ( $j=0,1,\dots,15$ )。已知如果算法2成功,则一定有 $n \leq 32$ ,由此可得 $r \leq 0$ 。容易看出式(7)的每一组非负整数解与同时满足式(2)和式(3)的一组解 $\{y_k\}$ 一一对应。所以同时满足式(2)和式(3)的解的数量等于式(7)的非负整数解的个数

$$s = \binom{r+15}{r}$$

因此当 $n$ 增大时, $r$ 同时增大,同时满足式(2)和式(3)的解的数量 $s$ 增多。所以说,只要引入的错误足够多,那么一定能得到整个 $M^9$ 。

如果通过算法2得到整个 $M^9$ ,那么由图1看到,利用对 $M$ 的正确密文 $C$ 可以立即得到 $K^{10}$ 。

#### 4 从 $K^{10}$ 到 $K^0$

根据AES的密钥调度算法,不难发现对AES-128来说,密钥调度是完全可逆的,即可以由 $K^{10}$ 逆向地推出初始密钥 $K^0$ 。文献[4]中的附录给出了这一过程的C伪代码。

还要指出,攻击算法也可以作用在AES-192和AES-256上,并可以得到 $M^{11}$ 和 $M^{14}$ ,从而得到相应的 $K^{12}$ 和 $K^{14}$ 。但是,由于AES-192和AES-256的密钥调度算法,仅仅有 $K^{12}$ 和 $K^{14}$ 还不能逆向地推出初始密钥 $K^0$ 。

#### 5 攻击的软件模拟

在Visual C++6.0开发环境下,对整个攻击进行了软件模拟。使用随机数生成函数产生0~127之间的随机整数来模拟向 $M^9$ 中随机引入比特错误。

从模拟的结果中看到,如果能向 $M^9$ 中反复随机地引入

140个比特错误,那么找到密钥的可能性将超过90%。C.Giraud在文献[3]中指出“引入50个比特错误找出密钥是可能的”,而从我们的模拟结果来看,如果只是随机地引入比特错误,不能更精确地控制错误的发生位置,那么只引入50个比特错误找出密钥的可能性不大。错误数量与成功概率的关系由图2给出。

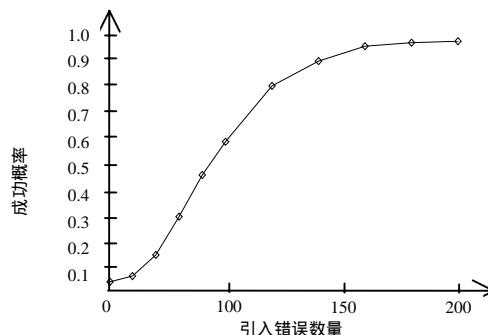


图2 错误数量与成功概率关系

软件模拟结果表明,如果在物理技术上能够很好地实现向 $M^9$ 中反复随机地引入比特错误,那么这种攻击方法是很有有效的。

#### 6 抵抗攻击

以密文分组链模式(CBC-Cipher Block Chaining)工作的AES可很好地抵抗本文提到的攻击。比特错误攻击的是利用

$$SB(x) \oplus SB(x \oplus e) = C_{SR(j)} \oplus F_{SR(j)}$$

来分别确定 $M^9$ 的每一个字节。这需要同一个明文分组 $M$ 的多个错误密文和一个正确密文。但是在CBC模式下,每次加密之前会先产生随机初始值 $IV$ (128bits),然后加密 $IV \oplus M$ 得到密文 $C$ ,由于每次加密前产生的随机初始值 $IV$ 不同,因此每次加密得到的密文都不同,中间结果 $M^9$ 也不相同。这样,攻击者总是无法得到一个确定的 $M^9$ ,攻击失败。

#### 7 结束语

差分错误分析已成为检测智能卡安全的重要因素。本文给出了一种对AES-128的差分错误分析算法,分析了算法成功的概率,并对整个攻击过程进行了软件模拟。攻击算法和模拟结果在检测智能卡安全中具有一定参考价值。

#### 参考文献

- 1 Biham E, Shamir A. Differential Fault Analysis of Secret Key Cryptosystems[M]. Lecture Notes in Computer Science. Springer, 1997, 1294: 513-525.
- 2 National Institute of Standards and Technology. Advanced Encryption Standard(AES)[S]. Federal Information Processing Standards Publication 197, 2001-11-26.
- 3 Giraud C. DFA on AES[Z]. 2003. <http://eprint.iacr.org/>.
- 4 Dusart P, Letourneux G, Vivolo O. Differential Fault Analysis on AES[Z]. 2003. <http://eprint.iacr.org/>.
- 5 Skorobogatov S, Anderson R. Optical Fault Induction Attack[M]. Lecture Notes in Computer Science, Springer, 2003, 2523: 2-12.