物联网环境下LED轻量级密码算法的安全性分析

李 玮[1),2),3)] 谷大武[2)] 赵 辰[1)] 刘志强[2)] 刘 亚[2)]

[1)](东华大学计算机科学与技术学院 上海 201601)

[2)](上海交通大学计算机科学与工程系 上海 200240)

[3)](上海市信息安全综合管理技术研究重点实验室 上海 200240)

摘 要 LED算法是于2011年CHES会议中提出的一种新型轻量级密码算法,用于在物联网环境下保护RFID标签以及智能卡等设备的通信安全.文中提出并讨论了一种针对LED算法的差分故障攻击方法.该方法采用面向半字节的随机故障模型,通过在LED算法中导入故障,分别仅需要3个错误密文和6个错误密文,即可恢复LED算法的64 bit和128 bit原始密钥.实验结果表明,针对LED算法的差分故障攻击方法不仅扩展了故障诱导的攻击范围,而且提高了故障诱导的效率,减少了错误密文数,从而为故障攻击其它轻量级密码算法提供了一种通用的分析手段.

关键词 物联网;RFID;轻量级密码算法;LED;差分故障分析

中图法分类号 TP309 **DOI号**:10.3724/SP.J.1016.2012.00434

Security Analysis of the LED Lightweight Cipher in the Internet of Things

LI Wei[1),2),3)] GU Da-Wu[2)] ZHAO Chen[1)] LIU Zhi-Qiang[2)] LIU Ya[2)]

[1)](*School of Computer Science and Technology, Donghua University, Shanghai* 201601)

[2)](*Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai* 200240)

[3)](*Shanghai Key Laboratory of Integrate Administration Technologies for Information Security, Shanghai* 200240)

Abstract LED, proposed in CHES 2011, is a new lightweight cipher which is applied in the Internet of Things to provide security for RFID tags and smart cards etc. On the basis of the half byte-oriented fault model and the differential analysis, a differential fault analysis on the LED cipher by inducing faults is proposed. Simulating experiment shows that our attack could recover its 64-bit secret key by introducing 3 faulty ciphertexts, and recover 128-bit secret key by introducing 6 faulty ciphertexts, respectively. Our method extends the attacking scope of faults, thereby increasing the efficiency of fault injection and decreasing the number of faulty ciphertexts. Thus, the results in this study will also be beneficial to the analysis of other lightweight ciphers.

Keywords Internet of Things; RFID; lightweight cipher algorithm; LED; Differential fault analysis

1 引 言

随着信息技术、计算机技术以及微电子技术的高速发展,物联网作为新一代信息化浪潮的典型代表,正逐步深入到人们生活的各个领域中,例如智能交通、现代物流、食品安全以及环境监测等.然而,物联网领域日益突出的数据安全问题使用传统密码算

收稿日期:2011-08-31;最终修改稿收到日期:2011-12-23.本课题得到国家自然科学基金(61003278,61073150)、上海市信息安全综合管理技术研究重点实验室开放课题资助基金、中央高校基本科研业务费专项资金资助.**李 玮**,女,1980年生,博士,讲师,主要研究方向为物联网安全与信息安全.E-mail:liwei.cs.cn@gmail.com.**谷大武**,男,1970年生,博士,教授,博士生导师,主要研究领域为信息安全.**赵 辰**,男,1987年生,硕士研究生,主要研究方向为计算机安全.**刘志强**,男,1977年生,博士研究生,主要研究方向为信息安全.**刘 亚**,女,1983年生,博士研究生,主要研究方向为计算机安全.

法却无法很好地解决，主要原因是物联网上使用的应用组件不同于传统的台式机和高性能计算机，是计算能力相对较弱的微型计算处理设备，运算存储能力有限. 所以，数据安全受到极大威胁. 在这种情况下，轻量级密码算法受到人们的广泛关注. 与传统密码算法相比，轻量级密码算法的执行效率更高、计算资源消耗更少、适合计算能力有限的 RFID (Radio Frequency Identification，射频识别)标签等微型计算设备使用. 在 2011 年召开的 CHES 会议上 Guo 等人[1]提出的 LED 算法就是这样一种典型的轻量级密码算法，与其它轻量级密码算法相比，它的软硬件执行效率更高、适应环境能力更强，是轻量级密码算法中的佼佼者.

在物联网环境下，轻量级密码算法的许多硬件实现和以硬件为表现形式的软件实现环境中，譬如 RFID 标签、智能卡、密码协处理器、密码芯片等，攻击者往往不仅具备传统密码攻击的条件，而且还可以通过电路剖析和软件逆向分析等手段获得算法的硬件实现结构和编码实现方法，并可以观察和测量密码变换信息，“干预”密码变换的正常运行使其出错. 攻击者利用这些额外出错信息有可能实现比“黑盒攻击”更有效的密码破译. 人们通常把这种环境下的攻击称为“故障攻击(Fault Analysis)”[2]. 在故障攻击中，差分故障攻击(Differential Fault Analysis，DFA)较容易实施且最难防御，攻击者通常只需获得密码器件的几个或几十个故障密文，就能以很低的计算量和存储量达到很高的成功率[3]. 因此，差分故障攻击凭借其成功的攻击效果和潜在的发展前景，已成为密码分析和密码工程领域发展最为迅速的方向之一.

目前，LED 算法的安全性研究仅限于传统密码分析，例如差分分析、线性分析和代数攻击等[1]. 在抗差分故障攻击的安全性方面，国内外尚未发现有公开发表的结果. 因此，研究 LED 算法的安全性，提出有效的差分故障攻击方法，对于保护物联网的通信安全，对于增强密码设备的自主设计、开发和分析，无疑具有重要意义.

在深入分析和测试 LED 算法后，本文提出了一种针对 LED 算法的高效差分故障攻击方法. 该方法针对物联网的环境特性，结合了随机故障模型和差分分析，以较少的故障密文数即可恢复原始密钥. 结果表明，本文提出的方法仅需要 3 个和 6 个错误密文即可分别破译 LED 算法的 64 比特和 128 比特原始密钥.

本文第 2 节介绍差分故障攻击的相关工作；第 3 节简单介绍 LED 算法；第 4 节提出差分故障分析 LED 算法的方法；第 5 节针对不同故障模型下，给出扩展的差分故障攻击方法；第 6 节给出攻击的计算复杂度分析；第 7 节介绍实验及相关结果分析；第 8 节总结并指出下一步的研究方向.

2　相关工作

差分故障攻击方法是一种新型的密码分析方法. 1997 年，Biham 和 Shamir[3]针对 DES 算法首次提出了差分故障攻击方法. 该方法先在算法运行时引入故障，使算法执行某些错误的操作、过程，并产生错误的结果，再利用这些结果，结合算法自身的数学特性和差分分析，破译出算法的关键信息(密钥). 后来根据不同的基本假设和故障模型，研究学者对 AES 算法进行了深入的研究[4-9]. DES 算法和 AES 算法的破译历程为其它密码的差分故障攻击奠定了坚实的基础. 后来，SMS4[10]、ARIA[11]、PRESENT[12]、KEELOG[13]等算法相继被验证具有差分故障攻击的隐患.

从差分故障攻击方法的发展情况来看，成功实现该方法需具备两个重要条件，即基本假设和故障模型. 基本假设是差分故障攻击成功实现的外部条件. 因为差分故障攻击方法与密码算法的实现环境密切相关，所以分析人员需要能够运行包含密码算法的实体(软件、设备、芯片等)，并能够选择和记录实体的输入和输出情况. 选择明文攻击是一种常被采用的基本假设.

此外，导入理想的故障，实现合适的故障模型，是差分故障攻击成功实现的内部条件. 通常，一个故障模型由三元组(时刻、位置、动作)构成：

(1) 故障时刻. 攻击需要精确控制错误发生的时刻，例如在运算进行时，将故障引入到某一个范围内(某些轮).

(2) 故障位置. 攻击需要精确控制错误发生的位置，例如将故障引入到某一指定的记忆单元，包括寄存器的某些位(或半字节、字节).

(3) 故障动作. 攻击需要精确控制错误发生的行为，通常分为设定故障位置的值取反、设定故障位置的值为预定值(已知故障值)、随机设定故障的值(未知故障值).

在基本假设一定的情况下，故障模型的选择范围越宽，则说明此分析技术适用范围越广. 在基本假

设和故障模型一定的情况下,攻击所需要的故障密文数越少,说明此分析技术的攻击效果越好.

因此,在面向密码算法的差分故障攻击方法中,国内外已有的研究工作主要集中在针对一些典型密码算法的安全性实现上,但是由于一些新型轻量级密码算法具有适用于物联网环境中的独特设计结构,其抗差分故障攻击的安全性还尚未被研究.针对这些新型算法,例如 LED 算法,结合基本假设和故障模型,提出一种适用范围广并且攻击效果好的差分故障攻击方法,对物联网环境下密码算法的安全实现和数据安全,具有重要意义.

3 LED 算法简述

LED 是一种具有 SPN(Substitution-Permutation Network,代换置换网络)结构的迭代型分组密码算法,分组长度为 64 bit,密钥长度分别为 64 bit 或 128 bit.根据密钥长度的不同,加密过程对应的轮数分别为 32 轮和 48 轮,表示为 LED-64 和 LED-128,如表 1 所示.由于解密过程与加密过程的结构相同,并且子密钥的使用顺序相同,因此本文仅介绍加密过程和密钥编排方案.

表 1 LED 算法的轮数和密钥长度的关系

密码算法	分组长度/bit	密钥长度/bit	轮数
LED-64	64	64	32
LED-128	64	128	48

3.1 术语说明

用 Z_2^e 表示 e bit 的向量集,Z_2^8 中的元素称为字节,Z_2^4 中的元素称为半字节.LED 算法中使用的基本运算主要有异或(⊕)、乘法(·)和连接(‖).

3.2 加密过程

LED 算法的加密过程如图 1 所示.设明文输入为 $X=(X_0,X_1,\cdots,X_{15})\in(Z_2^4)^{16}$,密文输出为 $Y=(Y_0,Y_1,\cdots,Y_{15})\in(Z_2^4)^{16}$,原始密钥 $K\in(Z_2^4)^{16}$,子密钥 $K_1\in(Z_2^4)^{16}$ 和 $K_2\in(Z_2^4)^{16}$.

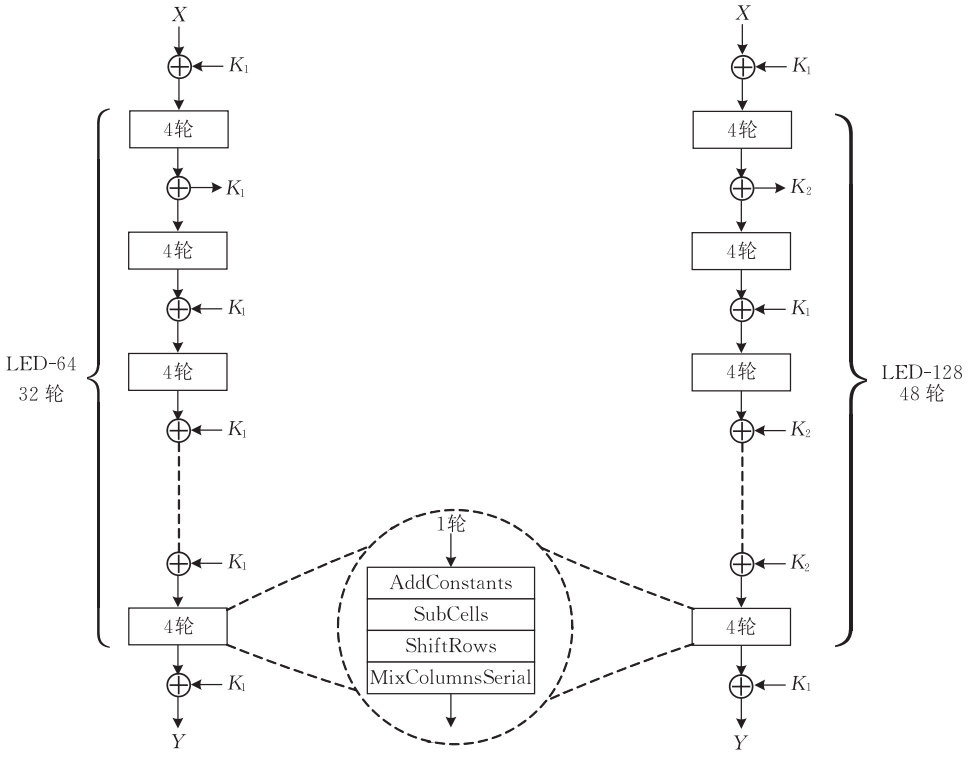

图 1 LED 算法的加密过程

加密过程包含了以下 5 个基本运算:

(1) AddRoundkey(子密钥加变换).中间状态与子密钥进行异或.

(2) AddConstants(常数相加变换).中间状态与常数相加.

(3) SubCells(信元代替变换).将中间状态中的每 4 bit 通过 S 盒非线性地变换为另外一个半字节.

(4) ShiftRows(行移位变换).对一个状态的每一行循环不同的位移量.第零行移位保持不变,第 1 行循环左移 4 bit,第 2 行循环左移 8 bit,第 3 行循环左移 12 bit.

(5) MixColumnsSerial(列混合变换).对一个

状态逐列进行变换，通过与矩阵

$$\begin{pmatrix} 4 & 2 & 1 & 1 \\ 8 & 6 & 5 & 6 \\ B & E & A & F \\ 2 & 2 & F & B \end{pmatrix}$$

相乘实现.

LED-64 和 LED-128 的加密过程分别如算法 1、算法 2 所示，轮数 l 分别为 32 轮、48 轮.

算法 1.　LED-64 加密过程.

输入：X，K
输出：Y
　$STATE=X$；
　for $a=1$ to 8 do
　　$STATE=$AddRoundKey($STATE$, K_1)；
　　for $b=1$ to 4 do
　　　$STATE=$MixColumnsSerial(ShiftRows
　　　　(SubCells(AddConstants($STATE$))))；
　　end for
　end for
　$Y=$AddRoundKey($STATE$,K_1).

算法 2.　LED-128 加密过程.

输入：X，K
输出：Y
　$STATE=X$；
　for $a=1$ to 6 do
　　$STATE=$AddRoundKey($STATE$, K_1)；
　　for $b=1$ to 4 do
　　　$STATE=$MixColumnsSerial(ShiftRows(SubCells
　　　　(AddConstants($STATE$))))；
　　end for
　　$STATE=$AddRoundKey($STATE$, K_2)；
　　for $b=1$ to 4 do
　　　$STATE=$MixColumnsSerial(ShiftRows(SubCells
　　　　(AddConstants($STATE$))))；
　　end for
　end for
　$Y=$AddRoundKey($STATE$,K_1).

3.3　密钥编排方案

在 LED-64 算法中，K 通过密钥编排方案生成了 K_1，它们的关系为

$$K=K_1.$$

在 LED-128 算法中，K 通过密钥编排方案生成了 K_1 和 K_2，它们的关系为

$$K=K_1 \| K_2.$$

4　LED 密码的差分故障攻击方法

4.1　基本记号和符号

记 $X\in(Z_2^4)^{16}$ 为明文输入，Y、Y^*、$\Delta Y\in(Z_2^4)^{16}$ 分别为密文、错误密文、差分密文，$K\in(Z_2^4)^{16}$ 为原始密钥，K_1、$K_2\in(Z_2^4)^{16}$ 为子密钥.

记 MC^{r-1} 为第 r 轮 AddConstants 运算的输入，AC^r 为第 r 轮 SubCells 运算的输入，SC^r 为第 r 轮 ShiftRows 运算的输入，SR^r 为第 r 轮 Mix-ColumnsSerial 运算的输入，其中，$1\le r\le l$.

记 ΔMC^{r-1} 为第 r 轮 AddConstants 运算的差分输入，ΔAC^r 为第 r 轮 SubCells 运算的差分输入，ΔSC^r 为第 r 轮 ShiftRows 运算的差分输入，ΔSR^r 为第 r 轮 MixColumnsSerial 运算的差分输入，其中，$1\le r\le l$.

记 AddConstants^{-1}、SubCells^{-1}、ShiftRows^{-1} 和 MixColumnsSerial^{-1} 分别为 AddConstants 运算、SubCells 运算、ShiftRows 运算和 MixColumnsSerial 运算的逆运算.

此外，第 r 轮 SubCells 运算的差分输入和差分输出满足

$$IN(\Delta AC_j^r,\Delta SC_j^r)=\{AC_j^r \mid AC_j^r\in Z_2^4,\ S(AC_j^r)\oplus S(AC_j^r\oplus\Delta AC_j^r)=\Delta SC_j^r,\ 0\le j\le 15,1\le r\le l\},$$

其中，S 表示 4×4 的 S 盒变换.

4.2　基本假设和故障模型

本文采用的差分故障攻击的基本假设为：对同一个明文，我们可以获得在同一个密钥作用下的正确密文和错误密文.

LED 算法采用半字节为基本单位的运算，因此，我们采用面向半字节的随机故障模型，诱发加密算法中单轮运算存储的单元发生 4 bit 的随机错误，而且无需知道具体错误位置和错误值. 在本节中，单轮运算的轮数指 LED-64 算法中的倒数第 3 轮，或 LED-128 算法中的倒数第 3 轮和倒数第 7 轮.

4.3　基本步骤

1. 选择任一明文进行加密，获得该明文对应的正确密文.

2. 当 LED 算法的加密过程运行到倒数第 3 轮运算时，在此轮进行故障诱导，并获得错误密文. 结合差分分析，能够恢复出该轮子密钥的部分信息；重复这一环节，直至完全恢复出子密钥 K_1. 如果原始密钥为 64 bit，转向步 4；否则，转向步 3.

3. 选择相同明文和原始密钥再次进行加密，当加密过程运行到倒数第 7 轮时，对此轮进行随机故障诱导，从而获得错误密文. 然后，利用步 2 中已经恢复出的子密钥 K_1，对最后 4 轮进行解密；由解密得到的中间值，结合差分分析，恢复出该轮子密钥的部分字节信息；重复这一过程，直至完全恢复出子密钥 K_2.

4. 使用已经恢复出子密钥，根据密钥编排方案，计算出

原始密钥的值.

4.4 具体过程

1. 随机选择一个明文 X,获得在原始密钥 K 作用下的正确密文 Y.

2. 恢复子密钥 K_1,过程如下:

对明文 X 在原始密钥 K 作用下再次加密,当算法运行到倒数第 3 轮的 AddConstants(或 SubCells、ShiftRows、MixColumnsSerial)运算时,诱导其输入发生 4 bit 的随机故障,由此得到的故障密文记为 Y^*,则计算差分密文的值为

$$\Delta Y = Y \oplus Y^*.$$

故障导入后的扩散图如图 2 所示,则 ΔSC^l 为

$$\Delta MC^l = (Y \oplus K_1) \oplus (Y^* \oplus K_1) = \Delta Y,$$

$$\Delta SC^l = \text{ShiftRows}^{-1}(\text{MixColumnsSerial}^{-1}(\Delta MC^l)) = \text{ShiftRows}^{-1}(\text{MixColumnsSerial}^{-1}(\Delta Y)).$$

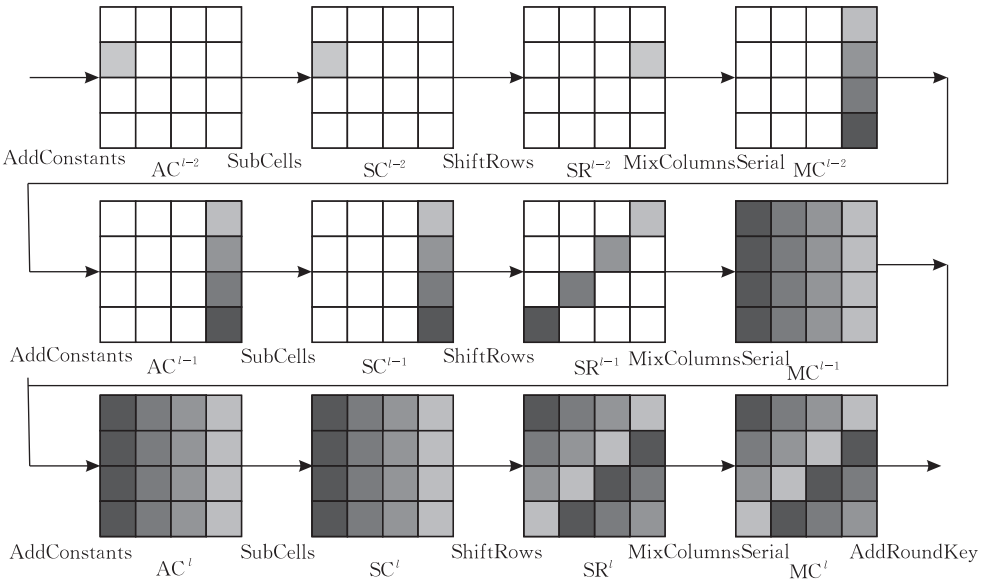

图 2 LED 算法最后 3 轮中的故障扩散路径

基于半字节随机故障的基本假设,计算 AC^l 的第 j 个字节值满足

$$S(AC_j^l) \oplus S(AC_j^l \oplus \Delta AC_j^l) = \Delta SC_j^l,$$

其中 $0 \le j \le 15$,ΔSC_j^l 为已知,$\Delta AC_j^l \in Z_2^4 \setminus \{0\}$. 重复此过程,通过收集相同明文和不同故障导致的多个故障密文,利用 ΔSC_j^l 和 ΔAC_j^l 的关系,循环运算取 AC^l 的交集,直至恢复出 AC^l 的唯一值. 在实际运算时,仅根据上述公式进行推导,AC_j^l 的值难以唯一确定,且计算复杂度较高. 在后文 4.5 节中,我们提出了一种高效方法推导出 AC^l 的唯一值. 在 AC^l 的唯一值确定后,计算 MC^l 的值:

$$MC^l = \text{MixColumnsSerial}(\text{ShiftRows}(\text{SubCells}(AC^l))),$$

因此,K_1 的值为

$$K_1 = Y \oplus MC^l.$$

3. 如果原始密钥长度为 128 bit,需对明文 X 在密钥 K 作用下再加密,当算法运行到倒数第 7 轮(即第 42 轮)时,诱导其输入产生 4 bit 的随机故障. 根据步 2 推导出的 K_1 直接解密最后 4 轮运算,获得倒数第 5 轮(即第 44 轮)的输出值为 $\text{AddConstants}^{-1}(AC^{l-3})$,则推导出倒数第 5 轮 SubCells 运算的差分输出值为

$$\Delta SC^{l-4} = \text{ShiftRows}^{-1}(\text{MixColumnsSerial}^{-1}(\Delta MC^{l-4})) = \text{ShiftRows}^{-1}(\text{MixColumnsSerial}^{-1}(\text{AddConstants}^{-1}(\Delta AC^{l-3}))).$$

利用关系:

$$S(AC_j^{l-4}) \oplus S(AC_j^{l-4} \oplus \Delta AC_j^{l-4}) = \Delta SC_j^{l-4},\ 0 \le j \le 15,$$

收集相同明文和不同故障产生的多个错误密文,利用 ΔSC^{l-4} 和 ΔAC^{l-4} 之间的关系,循环运算取 AC^{l-4} 的交集,直至恢复出 AC^{l-4} 的所有字节,进而推导出 K_2 的值:

$$K_2 = MC^{l-4} \oplus \text{AddConstants}^{-1}(AC^{l-3}) = \text{MixColumnsSerial}(\text{ShiftRows}(\text{SubCells}(AC^{l-4}))) \oplus \text{AddConstants}^{-1}(\text{SubCells}^{-1}(\text{ShiftRows}^{-1}(\text{MixColumnsSerial}^{-1}(\text{AddConstants}^{-1}(\text{SubCells}^{-1}(\text{ShiftRows}^{-1}(\text{MixColumnsSerial}^{-1}(\text{AddConstants}^{-1}(\text{SubCells}^{-1}(\text{ShiftRows}^{-1}(\text{MixColumnsSerial}^{-1}(\text{AddConstants}^{-1}(\text{SubCells}^{-1}(\text{ShiftRows}^{-1}(\text{MixColumnsSerial}^{-1}(Y \oplus K_1)))))))))))))))).$$

4. 根据密钥编排方案,计算出原始密钥 K 的值. 在 LED-64 算法中,$K = K_1$;在 LED-128 算法中,$K = K_1 \| K_2$.

4.5 推导 AC^l 的高效方法

在诱导故障的过程中,我们发现 4 bit 随机故障值通过第 $l-2$ 轮 SubCells、ShiftRows 和 MixColumnsSerial 的运算后,会影响到下一轮 AddConstants 输入的 16 bit,这 16 bit 继续影响到密文输出的全部字节. 在步 2 中,为了恢复出子密钥的值,如果直接采用穷尽搜索 AC^l 的方法,计算复杂度将达到 2^{64},并且难以恢复出它的唯一值. 这样的穷尽搜索方法在实际中将降低攻击的速度. 因此,本节提出一种有效的运算方法,充分利用故障扩散的关系来计算 AC^l.

在已知差分密文的情况下,第 l 轮 SubCells 运

算满足以下差分关系

$$
\begin{aligned}
&(\Delta Y_0,\Delta Y_7,\Delta Y_{10},\Delta Y_{13})=\\
&\quad(S(AC_0^l)\oplus S(AC_0^l\oplus\Delta AC_0^l),\\
&\quad S(AC_4^l)\oplus S(AC_4^l\oplus\Delta AC_4^l),\\
&\quad S(AC_8^l)\oplus S(AC_8^l\oplus\Delta AC_8^l),\\
&\quad S(AC_{12}^l)\oplus S(AC_{12}^l\oplus\Delta AC_{12}^l)),\\
&(\Delta Y_1,\Delta Y_4,\Delta Y_{11},\Delta Y_{14})=\\
&\quad(S(AC_1^l)\oplus S(AC_1^l\oplus\Delta AC_1^l),\\
&\quad S(AC_5^l)\oplus S(AC_5^l\oplus\Delta AC_5^l),\\
&\quad S(AC_9^l)\oplus S(AC_9^l\oplus\Delta AC_9^l),\\
&\quad S(AC_{13}^l)\oplus S(AC_{13}^l\oplus\Delta AC_{13}^l)),\\
&(\Delta Y_2,\Delta Y_5,\Delta Y_8,\Delta Y_{15})=\\
&\quad(S(AC_2^l)\oplus S(AC_2^l\oplus\Delta AC_2^l),\\
&\quad S(AC_6^l)\oplus S(AC_6^l\oplus\Delta AC_6^l),\\
&\quad S(AC_{10}^l)\oplus S(AC_{10}^l\oplus\Delta AC_{10}^l),\\
&\quad S(AC_{14}^l)\oplus S(AC_{14}^l\oplus\Delta AC_{14}^l)),\\
&(\Delta Y_3,\Delta Y_6,\Delta Y_9,\Delta Y_{12})=\\
&\quad(S(AC_3^l)\oplus S(AC_3^l\oplus\Delta AC_3^l),\\
&\quad S(AC_7^l)\oplus S(AC_7^l\oplus\Delta AC_7^l),\\
&\quad S(AC_{11}^l)\oplus S(AC_{11}^l\oplus\Delta AC_{11}^l),\\
&\quad S(AC_{15}^l)\oplus S(AC_{15}^l\oplus\Delta AC_{15}^l)).
\end{aligned}
$$

如图 2 所示，在第 $l-2$ 轮 AddConstants 运算中诱导半字节的随机故障后，我们观察到加密过程中含有一个重要的特性：ΔAC^l 中包含 16 个非零半字节差分值，这 16 个值是由 ΔAC^{l-1} 中的 4 个非零半字节差分值扩散而成，并且 ΔAC^{l-1} 中的 4 个非零半字节差分值是由 ΔAC^{l-2} 中的随机半字节故障值扩散而成，因此，ΔAC^l 的值满足以下关系式(图 3)：

$$
\Delta AC^l=\begin{pmatrix}\Delta AC_0^l & \Delta AC_1^l & \Delta AC_2^l & \Delta AC_3^l\\ \Delta AC_4^l & \Delta AC_5^l & \Delta AC_6^l & \Delta AC_7^l\\ \Delta AC_8^l & \Delta AC_9^l & \Delta AC_{10}^l & \Delta AC_{11}^l\\ \Delta AC_{12}^l & \Delta AC_{13}^l & \Delta AC_{14}^l & \Delta AC_{15}^l\end{pmatrix}\in
$$

$$
\left\{\begin{pmatrix}4&2&1&1\\8&6&5&6\\B&E&A&F\\2&2&F&B\end{pmatrix}\cdot\begin{pmatrix}p&0&0&0\\0&0&0&q\\0&0&u&0\\0&v&0&0\end{pmatrix},\right.
$$

$$
\begin{pmatrix}4&2&1&1\\8&6&5&6\\B&E&A&F\\2&2&F&B\end{pmatrix}\cdot\begin{pmatrix}0&p&0&0\\q&0&0&0\\0&0&0&u\\0&0&v&0\end{pmatrix},
$$

$$
\begin{pmatrix}4&2&1&1\\8&6&5&6\\B&E&A&F\\2&2&F&B\end{pmatrix}\cdot\begin{pmatrix}0&0&p&0\\0&q&0&0\\u&0&0&0\\0&0&0&v\end{pmatrix},
$$

$$
\left.\begin{pmatrix}4&2&1&1\\8&6&5&6\\B&E&A&F\\2&2&F&B\end{pmatrix}\cdot\begin{pmatrix}0&0&0&p\\0&0&q&0\\0&u&0&0\\v&0&0&0\end{pmatrix},\right.
$$

$$
\left.p,q,u,v\in Z_2^4,\ p、q、u、v\text{ 不同时为零}\right\}.
$$

$$
=\left\{\begin{pmatrix}4p&v&u&2q\\8p&6v&5u&6q\\Bp&9v&Au&Eq\\8p&Bv&Fu&2q\end{pmatrix},\begin{pmatrix}2q&4p&v&u\\6q&8p&6v&5u\\Eq&Bp&9v&Au\\2q&2p&Bv&Fu\end{pmatrix},\right.
$$

$$
\begin{pmatrix}u&2q&4p&v\\5u&6q&8p&6v\\Au&Eq&Bp&9v\\Fu&2q&2p&bv\end{pmatrix},\begin{pmatrix}v&u&2q&4p\\6v&5u&6q&8p\\9v&Au&Eq&Bp\\Bv&Fu&2q&2p\end{pmatrix},
$$

$$
\left.p,q,u,v\in Z_2^4,\ p、q、u、v\text{ 不同时为零}\right\}.
$$

根据 SubCells 运算输入输出满足的差分关系，结合半字节故障扩散的路径，认为 ΔAC^l 的所有可能值最多为

$$2^8(=2^4\cdot 2^2\cdot 2^2),$$

其中，$2^4\cdot 2^2$ 表示第 $l-1$ 轮 ΔMC^{l-1} 四列的可能值范围(假设导入故障位置列数一定)，2^2 表示故障扩散存在四种情况，如图 3 所示. 由于 ΔMC^{l-1} 的取值范围与 ΔAC^l 的范围相同，且已知 ΔY 每一列值的情况下，可以逐列求出 AC^l 的值. 从而，一次故障导入所需的复杂度降低为

$$2^{24}(=2^{16}\cdot 2^4\cdot 2^2\cdot 2^2).$$

如果不利用上述 ΔAC^l 的特性，其取值范围为 2^{64}，搜索空间为全集，难以恢复唯一值.

最后，继续导入随机故障，重复上述运算，直至恢复出的 AC^l 中的每个元素仅为唯一值.

5　扩展的差分故障攻击方法

5.1　故障位置判断方法

基于以上对故障位置和错误密文数的分析，如

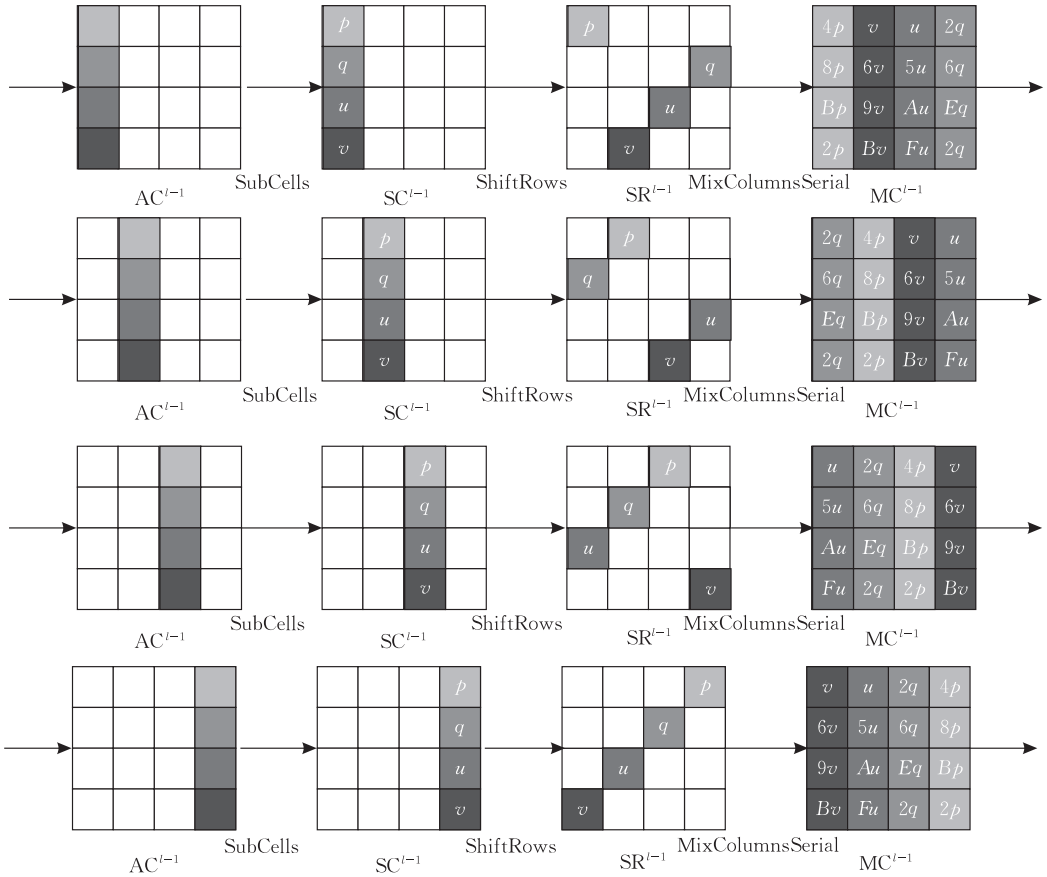

图 3 LED 算法故障扩散的 4 种情况

果步 2 中故障没有被导入指定位置(倒数第 3 轮),而是被导入到最后两轮运算中,这样做的结果必然降低了故障导入的效率,而且增加了恢复原始密钥所需的故障密文数. 因此,我们提出一种判断方法用以区分最后 3 轮运算的导入位置. 当随机故障被导入在加密算法中,密文的差分输出与故障的导入位置存在以下 3 种情况:

(1) 如果故障被导入在最后一轮运算中,则 AC^l 全部字节中仅发生 4 bit 变化,从而加密过程的 SubCells 输出全部字节中也只出现 4 bit 变化,记为 $\lambda_0 \in Z_2^4 \backslash \{0\}$. 因此,$\lambda_0$ 经过 ShiftRows 运算和 MixColumnsSerial 运算后,密文的差分输出值 ΔY 必满足下列值之一:

$\{$MixColumnsSerial(ShiftRows(λ_0,0,0,…,0)),
MixColumnsSerial(ShiftRows(0,λ_0,0,…,0)),
MixColumnsSerial(ShiftRows(0,0,λ_0,…,0)),…,
MixColumnsSerial(ShiftRows(0,0,0,…,λ_0)),
$\lambda_0 \in Z_2^4 \backslash \{0\}\}$.

经统计,共有 240(=15×16)个值,因此,密文的差分输出的所有字节出现规律性变化.

(2) 如果故障被导入在倒数第 2 轮运算中,则 AC^l 全部字节中仅发生 16 bit 变化,从而加密过程的 SubCells 运算输出的全部字节中也只含有 16 bit 变化,记为 $\{\lambda_1,\lambda_2,\lambda_3,\lambda_4\} \in \{Z_2^4\}^4 \backslash \{0\}$. 因此,$\{\lambda_1,\lambda_2,\lambda_3,\lambda_4\}$ 经过 ShiftRows 运算和 MixColumnsSerial 运算后,密文的差分输出值 ΔY 必属于下列集合:

$\{$MixColumnsSerial(ShiftRows(λ_1,0,0,0,
λ_2,0,0,0,λ_3,0,0,0,λ_4,0,0,0)),
MixColumnsSerial(ShiftRows(0,λ_1,0,0,0,
λ_2,0,0,0,λ_3,0,0,0,λ_4,0,0)),
MixColumnsSerial(ShiftRows(0,0,λ_1,0,0,0,
λ_2,0,0,0,λ_3,0,0,0,λ_4,0)),
MixColumnsSerial(ShiftRows(0,0,0,λ_1,0,
0,0,λ_2,0,0,0,λ_3,0,0,0,λ_4)),$\{\lambda_1,\lambda_2,\lambda_3,\lambda_4\} \in \{Z_2^4\}^4 \backslash \{0\}\}$.

经统计,共有 262140(=65535×4)个值,因此,密文的差分输出的所有字节出现规律性变化.

(3) 如果故障被导入在倒数第 3 轮运算中,则 AC^l 全部字节中会发生 64 bit 变化,从而加密过程的 SubCells 运算的输出字节全部发生变化,密文的差分输出值均没有上述规律性变化.

因此,为了辨别故障导入的存储单元,仅需计

算正确密文 Y 和错误密文 Y^* 的差分输出 ΔY. 在基本假设中，当故障位置被诱导在加密算法最后 3 轮中任意存储中间单元时，如果 ΔY 的值属于 262380(=240+262140)个值中的一个，则说明故障导入的位置在存储单元最后两轮中，否则故障导入的位置在倒数第 3 轮中. 这种方法简单易用，可以有效地提高故障导入的效率，并且可以降低恢复子密钥时所需的错误密文数.

5.2　扩展的故障模型

通过采用面向半字节的随机故障模型，分析人员每次诱发单轮运算存储值发生 4 bit 的随机错误，并且无需知道具体错误位置和错误值. 此单轮运算指 LED-64 算法中的倒数第 3 轮，或 LED-128 算法中的倒数第 3 轮和倒数第 7 轮. 在物联网实际环境中，这一故障模型范围较窄，在实际中攻击者具有将故障导入到其它轮位置的能力. 因此，我们利用 5.1 节故障位置分布规律，将故障模型中的单轮运算扩展为以下情况：

(1) LED-64 算法中的倒数第 3 轮，或 LED-128 算法中的倒数第 3 轮和倒数第 7 轮.

(2) LED-64 算法中的倒数第 2 轮，或 LED-128 算法中的倒数第 2 轮和倒数第 6 轮.

(3) LED-64 算法中的倒数第 1 轮，或 LED-128 算法中的倒数第 1 轮和倒数第 5 轮.

值得注意的是，在情况(1)(2)下，差分故障分析可以成功实现并恢复子密钥；在情况(3)下，在没有故障其它信息的帮助下(例如，已知故障的汉明重量)，我们依靠差分故障攻击难以直接恢复子密钥，仅能判断故障导入位置.

5.3　扩展攻击的基本步骤

分析人员可以实际条件的不同，选择合适的故障模型. 在扩展的故障模型下，攻击方法仍与原方法类似，仅需在子密钥恢复前对故障位置进行判断，从而采取性能更好的差分故障攻击方法. 总体而言，基本过程如下：

1. 选择明文攻击，获得该明文对应的正确密文.

2. 在算法加密过程中进行故障诱导，根据差分密文的结构判断出故障导入轮数. 如果故障导入轮数不合适，重新诱导故障，直至合适故障出现，并获得所需要的错误密文. 利用差分分析，恢复出该轮子密钥的部分字节信息；重复这一过程，直至完全恢复出子密钥 K_1. 如果原始密钥为 64 bit，转向步 4；否则，转向步 3.

3. 利用步 2 中已经恢复出的子密钥 K_1，对最后 4 轮进行解密. 在加密过程的倒数第 5 轮运算之前进行故障诱导，根据倒数第 5 轮差分输出的结构判断故障导入轮数. 如果故障导入轮数不合适，重新诱导故障，直至合适故障出现，并获得所需要的错误输出. 由解密得到的中间值，结合差分分析，恢复出该轮子密钥的部分字节信息；重复这一过程，直至完全恢复出子密钥 K_2.

4. 使用已经恢复出子密钥，结合密钥编排方案，计算出原始密钥的值.

6　复杂度分析

在攻击过程中，恢复一个子密钥所需要的故障密文数由故障诱导的位置和故障模型决定. 在理想的故障模型下，如果故障发生在最后一轮的运算中，则非线性层的输入中仅有一个 4 bit 发生变化，说明通过差分故障攻击仅能恢复出该轮子密钥的 4 bit，为了恢复一个子密钥的所有字节，必须诱导大量的故障；如果 4 bit 故障发生在最后一轮之前的某一轮运算中，即该轮非线性层的差分输入和差分输出仅包含一个非零 4 bit 值，线性变换的扩散性使得其差分输出含有多个非零 4 bit 值. 因此，经过若干轮运算后，最后一轮非线性层的差分输入和差分输出包含多个非零 4 bit，依靠这种方法可以提高故障导入的效率. 在推导子密钥的交集推导运算中，至少需要两个故障导入，我们通过求候选子密钥的交集直至推导出包含唯一值，因此恢复一个子密钥所需的最少故障密文数理论值为

$$\begin{cases} 0, & d=0 \\ \left\lceil \dfrac{2n}{d} \right\rceil, & 1\leq d\leq n \end{cases},$$

其中，n 表示 SubCells 运算的输出比特数，d 表示通过两个故障密文可恢复出的子密钥比特数. 如果 $d=0$，则表明攻击没有恢复出子密钥的任何比特，因此最少故障密文数理论值为 0.

在诱导一个故障发生时，穷尽搜索所需的最大复杂度为

$$2^{x\cdot si}\cdot\left(2^{si}\cdot\left\lceil\frac{n}{si}\right\rceil\right),$$

其中，n 表示 SubCells 运算的输出比特数，si 表示一个 S 盒的输入比特数，x 表示使用 S 盒进行并行计算的个数.

在一个故障模型中，恢复原始密钥所需要的攻击复杂度为

$$\begin{cases} 0, & d=0 \\ 2^{(x+1)\cdot si+1}\cdot\left\lceil\dfrac{n^2\cdot g}{si\cdot d}\right\rceil, & 1\leq d\leq n \end{cases},$$

其中 si 为一个 S 盒的输入比特数，n 表示 SubCells

运算的输出比特数，d 表示通过故障可恢复出的子密钥比特平均数，g 表示恢复原始密钥所需要的最少子密钥数目.

以子密钥 K_1 的推导为例，本节说明攻击过程的复杂度与故障导入的位置直接相关. 在加密算法运算过程中，故障可以被随机地导入最后一轮、倒数第 2 轮和倒数第 3 轮运算中的存储单元中.

如果故障被导入在最后一轮运算中，那么加密过程中最后一轮 SubCells 的输入和输出仅发生 4 bit 的变化，此时差分故障分析难以直接恢复子密钥. 如果已知故障的某些信息，例如已知故障汉明重量，则差分故障分析可以成功实现. 在这种情况下，为了恢复出子密钥 K_1 的所有字节，必须在最后一轮运算中多次导入故障. 根据 SubCells 运算的差分关系，如果要成功恢复出子密钥的 4 个比特位，即恢复出一个 AC_j^l 的值，则分析过程至少需要 2 个不同的错误密文(故障发生在同一半字节位置，但其错误值可以是随机的，且已知故障汉明重量)即可. 因此，要想唯一地恢复出子密钥的全部字节，理论上至少需要 32 个不同的错误密文. 以此类推，要恢复出其它子密钥，理论上至少需要 32 个错误密文. 也就是说，要完全恢复 LED 算法的 64 bit 和 128 bit 原始密钥，理论上分别至少需要 32 个和 64 个错误密文，攻击所需的复杂度分别为

$$2^{29}\left(=2^{(4+1)\cdot 4+1}\cdot\frac{64^2\cdot 1}{4\cdot 4}\right)$$

和

$$2^{30}\left(=2^{(4+1)\cdot 4+1}\cdot\frac{64^2\cdot 2}{4\cdot 4}\right),$$

此时，$n=64$，$si=4$，$x=4$，$d=4$，$g=1$(LED-64 算法)或 $g=2$(LED-128 算法).

如果故障被导入在倒数第 2 轮运算中，以导致加密过程中最后一轮 SubCells 的输入和输出仅发生 16 bit 的变化，为了恢复出子密钥 K_1 的所有字节，必须在倒数第 2 轮运算中多次导入故障. 并且，要成功恢复出子密钥的某 16 bit，至少需要 2 个不同的错误密文(故障发生在同一半字节位置，但其错误值可以是随机的). 因此，唯一地恢复出子密钥的全部字节，理论上至少需要 8 个不同的错误密文. 以此类推，要恢复出其它子密钥，理论上也至少需要 8 个错误密文. 也就是说，要完全恢复 LED 算法的 64 bit 和 128 bit 原始密钥，理论上分别需要 8 个和 16 个错误密文，攻击所需的复杂度分别为

$$2^{27}\left(=2^{(4+1)\cdot 4+1}\cdot\frac{64^2\cdot 1}{4\cdot 16}\right)$$

和

$$2^{28}\left(=2^{(4+1)\cdot 4+1}\cdot\frac{64^2\cdot 2}{4\cdot 16}\right),$$

其中，$n=64$，$si=4$，$x=4$，$d=16$，$g=1$(LED-64 算法)或 $g=2$(LED-128 算法).

如果故障被导入在倒数第 3 轮运算中，则加密过程中最后一轮 SubCells 的输入输出均发生 16 个半字节的变化. 这样一次故障导入可以恢复出子密钥的 16 个半字节，要成功恢复出子密钥 K_1 的全部字节，则平均需要 2 个不同的错误密文(故障可发生在任一字节位置，且其错误值可以是随机的). 以此类推，要恢复出其它子密钥，理论上只需要 2 个错误密文. 也就是说，要完全恢复 LED 算法的 64 bit 和 128 bit 原始密钥，理论上分别需要 2 个和 4 个错误密文，攻击所需的复杂度分别为

$$2^{25}\left(=2^{(4+1)\cdot 4+1}\cdot\frac{64^2\cdot 1}{4\cdot 64}\right)$$

和

$$2^{26}\left(=2^{(4+1)\cdot 4+1}\cdot\frac{64^2\cdot 2}{4\cdot 64}\right),$$

其中，$n=64$，$si=4$，$x=4$，$d=64$，$g=1$(LED-64 算法)或 $g=2$(LED-128 算法).

7 攻击实验和结果分析

在普通 PC 机器(CPU 为 AMD sempron(tm) processor 2600+1.60 GHz，内存 2 GB)上我们使用 C 语言编程实现了本文给出的攻击方法，其中故障诱导得到错误密文的过程用计算机软件模拟，完成了 1000 组实验.

在一定的基本假设下，我们针对不同的故障模型，用以恢复 LED 算法的密钥. 实验基本过程如下：

首先，选定一个故障模型，即故障的时刻、位置、动作分别为加密算法倒数第 3 轮(或倒数第 2 轮)运算、4 bit 故障、随机未知故障值；或加密算法最后一轮运算、4 bit 故障、限定未知故障值；

其次，在故障模型下，通过选择明文及相应的正确和故障密文对，对密码算法的执行过程进行差分分析，给出计算子密钥信息的步骤和数学表达式；

最后，结合故障位置判断方法，利用计算机程序模拟该故障模型，对实验中得到的数据进行数学处理和计算，求得若干子密钥，结合密钥编排方案求出原始密钥.

以 LED-64 子密钥恢复为例. 为了恢复子密钥的唯一值，我们通过收集相同明文和不同故障导致

的多个故障密文，运算取 AC^l 的交集，直至恢复出 AC^l 的所有字节. 在实验中，我们采用并行恢复 AC^l 的四列值的方式，基于一个正确密文和一个故障密文时，AC^l 的每一列可能侯选值个数均值为 29.4、25.5、28.7 以及 18.3；基于一个正确密文和两个故障密文时，AC^l 的每一列可能侯选值相交个数均值为 1.0、1.6、1.9、1.3；基于一个正确密文和两个故障密文时，AC^l 的每一列侯选值个数均值为 0.6、0.5、0.6 以及 0.6. 以上数据表明，仅依靠一次故障导入无法推导出 AC^l 任一列的唯一值，也就无法推导 AC^l 的唯一值，即子密钥的唯一值.

因此，在差分故障攻击的分析过程中，通常需要使用至少两次故障导入才能恢复子密钥的唯一值，结合 LED 算法的特性，实验表明：在半字节故障模型下，在第 2 次故障导入时，利用两次故障求出的候选子密钥集合进行相交，分布如图 4 所示，其中数目为 1 的比例为 58%，说明子密钥已经被唯一恢复出来的成功率为 58%；数目为 2 及以上的比例为 42%，说明其余的子密钥仍存在多个候选值；继续进行第 3 次故障导入，分布如图 5 所示，其中数目为 1 的比例为 92%，说明子密钥已经被唯一恢复出来的成功率为 92%；数目为 2 及以上的比例为 8%，说明其余子密钥仍存在多个候选值. 从以上实验推出，如果仅进行两次故障导入，唯一恢复子密钥的成功率为 58%；如果进行三次故障导入，唯一恢复子密钥的成功率为 92%.

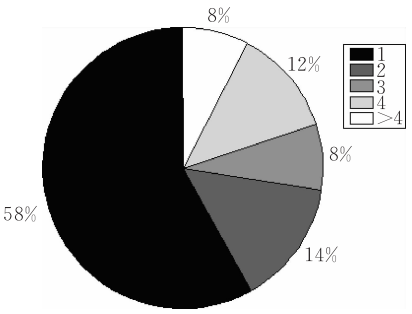

图 4　第 2 次故障导入候选子密钥数分布图

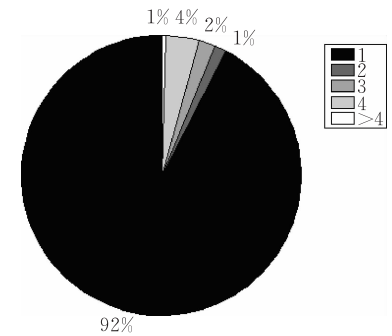

图 5　第 3 次故障导入候选子密钥数分布图

研究结果说明，LED 密码不能抵抗差分故障攻击. 在面向 4 比特的随机故障模型下，分析人员可以通过差分密文的特性，检测并判断合适的故障位置. 实验表明，每次可以诱发单轮存储中间值发生错误，在恢复子密钥成功率较高的情况下，恢复 64 bit 密钥最少仅需要 3 个故障密文，恢复 128 bit 密钥最少仅需要 6 个故障密文，耗时不超过 1 s. 此方法针对物联网环境中的不同设备，因此适应性广，不仅扩大了故障导入范围，而且提高了故障诱导的效率，如表 2 和表 3 所示.

表 2　LED-64 算法中不同故障模型下，错误密文数和计算复杂度的关系

故障模型	理论错误密文数	实际错误密文数	理论计算复杂度	实际计算复杂度
第 32 轮、4 bit、限定	32	48	2^{29}	2^{30}
第 31 轮、4 bit、未知	8	12	2^{27}	2^{28}
第 30 轮、4 bit、未知	2	3	2^{25}	2^{26}

表 3　LED-128 算法中不同故障模型下，错误密文数和计算复杂度的关系

故障模型	理论错误密文数	实际错误密文数	理论计算复杂度	实际计算复杂度
第 48 轮和第 42 轮、4 bit、限定	64	92	2^{30}	2^{31}
第 47 轮和第 43 轮、4 bit、未知	16	24	2^{28}	2^{29}
第 46 轮和第 44 轮、4 bit、未知	4	6	2^{26}	2^{27}

8　结束语

本文提出并讨论了 LED 算法抗差分故障攻击的安全性. 理论分析和实验结果表明，LED 算法易受差分故障攻击威胁. 在选择明文攻击下，通过扩展故障诱导的位置至加密算法的最后若干轮运算，能够以较少的错误密文数筛选出候选子密钥，进而恢复出 LED 算法的原始密钥，达到既扩展故障诱导的攻击范围，又减少错误密文数，提高故障诱导攻击成功率的目的. 此方法有助于优化差分故障攻击方法的攻击性能，提高故障攻击的效率和实用性.

下一步的研究方向是从故障诱导的随机性和通用算法结构等方面对差分故障攻击方法进行探索，采取措施优化其攻击性能，提高攻击的实用性和效率. 在后续的研究中，我们将扩展故障攻击的分析对象，并从软硬件设计和通用算法结构上对差分故障攻击进行探索.

致　谢　感谢本文审稿专家和编辑所提出的宝贵意见和建议！

参 考 文 献

[1] Guo J, Peyrin T, Poschmann A et al. The LED block cipher//Proceedings of the International Workshop of Cryptographic Hardware and Embedded Systems(CHES 2011). Nara, Japan, 2011: 326-341

[2] Boneh D, DeMillo R A, Lipton R J. On the importance of checking cryptographic protocols for faults//Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT 1997). Konstanz, Germany, 1997: 37-51

[3] Biham E, Shamir A. Differential fault analysis of secret key cryptosystems//Proceedings of the 17th Annual International Cryptology Conference(CRYPTO 1997). California, USA, 1997: 513-525

[4] Blömer J, Seifert J. Fault based cryptanalysis of the advanced encryption standard (AES 2004)//Proceedings of the 7th International Conference Financial Cryptography (FC 2003). Guadeloupe, French West Indies, 2003: 162-181

[5] Giraud C. DFA on AES//Proceedings of the 4th International Conference of Advanced Encryption Standard 4 (AES 2004). Bonn, Germany, 2005: 27-41

[6] Chen C, Yen S. Differential fault analysis on AES key schedule and some countermeasures//Proceedings of the Australasian Conference on Information Security and Privacy (ACISP 2003), Wollongong, Australia, 2003: 118-129

[7] Dusart P, Letourneux G, Vivolo O. Differential fault analysis on AES///Proceedings of the Applied Cryptography and Network Security (ACNS 2003). Kunming, China, 2003: 293-306

[8] Piret G, Quisquater J J. A differential fault attack technique against SPN structures, with application to the AES and KHAZAD//Proceedings of the International Workshop of Cryptographic Hardware and Embedded Systems (CHES 2003). Cologne, Germany, 2003: 77-88

[9] Amir M, Mohammad T M S, Mahmoud S. A generalized method of differential fault attack against AES cryptosystem///Proceedings of the International Workshop of Cryptographic Hardware and Embedded Systems (CHES 2003). Cologne, Germany, 2003: 91-100

[10] Zhang Lei, Wu Wen-Lin. Differential fault analysis on SMS4. Chinese Journal of Computers, 2006, 29(9): 1596-1602(in Chinese)
(张蕾, 吴文玲. SMS4 密码算法的差分故障攻击. 计算机学报, 2006, 29(9): 1596-1602)

[11] Li W, Gu D, Li J. Differential fault analysis on the ARIA algorithm. Information Science, 2008, 178(19): 3727-3737

[12] Li Juan-Ru, Gu Da-Wu. Differential fault attack on PRESENT block cipher//Proceedings of the 2009 Annual Conference of the Chinese Association for Cryptologic Research (CHINACRYPT 2009). Guangzhou, China, 2009: 3-13(in Chinese)
(李卷孺, 谷大武. PRESENT 算法的差分故障攻击//中国密码学会 2009 年会. 广州, 中国, 2009: 3-13)

[13] You Jian-Xiong, Li Rui-Lin, Li Chao. Fault attack on lightweight block cipher Keeloq. Acta Scientiarum Naturalium Universitatis Pekinensis, 2010, 46(5): 756-762(in Chinese)
(游建雄, 李瑞林, 李超. 轻量级分组密码 Keeloq 的故障攻击. 北京大学学报(自然科学版), 2010, 46(5): 756-762)

附录. 实验数据及结果.

任意选择明文和加密密钥如下:

以 LED-64 算法为例:

明文: 01 23 45 67 89 ab cd ef

密钥: 01 23 45 67 89 ab cd ef

攻击实验数据如下:

正确密文: fd d6 fb 98 45 f8 14 56

通过以下 3 种方式均可以恢复子密钥为

01 23 45 67 89 ab cd ef

对倒数第 3 轮的输入进行故障诱导得到的错误密文:

序号	错误密文
1	82 ee 75 9e 1c 58 a3 58
2	6c cb f4 7e f1 5a 96 ad
3	51 b8 ab 31 16 9a c1 61

对倒数第 2 轮的输入进行故障诱导得到的错误密文:

序号	错误密文
1	c6 9b 5e 5e 5a e4 78 0c
2	0b 89 b7 6b 12 3e c7 c3
3	bc 0d 30 1d 50 57 4a ca
4	a1 af 5c db 96 7f ab 97
5	4a 3d 05 81 c4 59 4e 1e
6	3b 8d 41 87 ad d0 7a 63
7	2c b7 b0 ea a0 de 2a 9f
8	a1 af 5c db 96 7f ab 97
9	e6 ef 9e c4 aa b3 38 21
10	dc 83 ad 37 eb 24 e6 ef
11	6e 81 75 46 ba c9 75 fc
12	27 07 92 0b 83 75 b3 aa

对倒数第 1 轮的输入进行故障诱导得到的错误密文:

序号	错误密文	序号	错误密文
1	bd d6 5b 98 35 f8 c4 56	25	ad d6 5b 98 95 f8 a4 56
2	f6 d6 fe 98 4a f8 18 56	26	f9 d6 f0 98 48 f8 1c 56
3	fd de fb 9b 45 fd 14 52	27	8d d6 6b 98 15 f8 34 56
4	fd d5 fb 92 45 f9 14 50	28	5d d6 8b 98 d5 f8 44 56
5	f7 d6 f9 98 41 f8 12 56	29	1d d6 eb 98 e5 f8 54 56
6	cd d6 5b 98 55 f8 74 56	30	fd d4 fb 94 45 f7 14 52
7	fd 76 fb 08 45 98 14 26	31	fd 66 fb c8 45 08 14 96
8	4d d6 6b 98 55 f8 94 56	32	6d d6 3b 98 f5 f8 64 56
9	f4 d6 f3 98 4a f8 12 56	33	fd c6 fb a8 45 28 14 96
10	ed d6 cb 98 95 f8 d4 56	34	fd f6 fb 58 45 08 14 16
11	9d d6 3b 98 35 f8 24 56	35	f4 d6 f3 98 4a f8 12 56
12	dd d6 bb 98 85 f8 04 56	36	4d d6 1b 98 25 f8 c4 56
13	fd 16 fb 28 45 18 14 36	37	fd d7 fb 9a 45 fe 14 5f
14	fa d6 f2 98 40 f8 16 56	38	bd d6 3b 98 55 f8 44 56
15	fd c6 fb 28 45 a8 14 b6	39	fd c6 fb 28 45 a8 14 b6
16	f1 d6 fc 98 46 f8 1b 56	40	fe d6 fd 98 4f f8 1c 56
17	ed d6 4b 98 15 f8 f4 56	41	f6 d6 fe 98 4a f8 18 56
18	f5 d6 fe 98 4c f8 17 56	42	f4 d6 f3 98 4a f8 12 56
19	fd e6 fb 78 45 08 14 46	43	fd 86 fb 38 45 28 14 e6
20	fd 86 fb 68 45 38 14 c6	44	f8 d6 f4 98 49 f8 1d 56
21	fd d9 fb 9b 45 fe 14 53	45	fd b6 fb 38 45 78 14 b6
22	6d d6 3b 98 f5 f8 64 56	46	ff d6 fd 98 4c f8 1f 56
23	fd d4 fb 9d 45 f2 14 59	47	f6 d6 f4 98 4d f8 11 56
24	f1 d6 f6 98 4c f8 10 56	48	fd d5 fb 9e 45 f2 14 5e

LI Wei, born in 1980, Ph. D., lecturer. Her research interests include IOT security and information security.

GU Da-Wu, born in 1970, Ph. D., professor, Ph. D. supervisor. His research interest is information security.

ZHAO Chen, born in 1987, M. S. candidate. His research interest is computer security.

LIU Zhi-Qiang, born in 1977, Ph. D. candidate. His research interest is information security.

LIU Ya, born in 1983, Ph. D. candidate. Her research interest is computer security.

Background

Our work is supported by the National Natural Science Foundation of China under Grant No. 61003278 and No. 61073150, the Opening Project of Shanghai Key Laboratory of Integrate Administration Technologies for Information Security, and the Fundamental Research Funds for the Central Universities. The projects aim to discuss the security of the lightweight ciphers against fault analysis.

A new lightweight cipher, called LED, was proposed in CHES 2011. It has better hardware and software performances, and is suitable in the Internet of Things. Up to now, the LED cipher is resistant to most classical attacks, such as differential analysis, linear analysis, etc. However, little attention has been paid to its security analysis against the differential fault analysis.

Differential fault analysis was first proposed by Biham and Shamir on DES in 1997. The similar analysis has been applied to some block ciphers, such as AES, ARIA, SMS4, PRESENT, KEELOG etc. Other than classical attacks, differential fault analysis is based on deriving information about the secret key by examining the differences between a cipher resulting from correct operation and a cipher of the same initial message resulting from faulty operation. Thus, this strong analysis could be applied into practice to break ciphers with less time and data complexity.

This paper presents an approach of differential fault analysis on the LED cipher. It makes use of the half byte-oriented fault model and the differential analysis. By assuming different round location of occurring fault, the 64-bit or 128-bit master key for the LED cipher could be obtained by 3 or 6 faulty ciphertexts. Thus, our method not only expands the location of occurring fault, but also improves the efficiency of the attack. It provides a general method for fault analysis on other lightweight ciphers.