针对流密码LEX的差分故障攻击及算法改进分析

李佳雨 石 会 邓元庆 龚 晶 关 宇
(解放军理工大学通信工程学院 南京 210007)

摘 要 分析了针对LEX算法的差分故障攻击。为增强LEX抗差分故障攻击的能力,采取将每组轮密钥异或一个128比特随机序列的方法,对其进行了改进。在此基础上,分析了改进算法的安全性和运算速度,并用一个实例仿真检验了改进算法的密钥流随机性。结果表明,改进的LEX算法能够抵抗差分故障攻击,并具备与原LEX算法相同的运算速度和密钥流随机性,提高了LEX算法的密码性能。

关键词 流密码,LEX,安全性,差分故障攻击,随机性
中图法分类号 TN918.4 **文献标识码** A

Differential Fault Attack and Analysis of Improvement on LEX

LI Jia-yu SHI Hui DENG Yuan-qing GONG Jing GUAN Yu
(College of Communications Engineering, PLA University of Science & Technology, Nanjing 210007, China)

Abstract A method of differential fault attack on LEX was analyzed. In order to enhance the ability of stream cipher LEX to resist differential fault attack, based on the idea of making every RoundKey xor a 128-bit random stream, a new version of LEX was proposed. Then the safety and speed of the improved LEX algorithm were analyzed, and an exam-ple was performed to test the randomicity of the improved algorithm's key stream. The results show that the improved algorithm is resistant to differential fault attack, and simultaneously, has the same computing speed and randomicity to the original LEX algorithm, which is an improvement of LEX.

Keywords Stream cipher, LEX, Security, Differential fault attack, Randomicity

1 引言

ECRYPT Ⅱ(European Network of Excellence for Cryptology Ⅱ)是欧洲委员会FP7(7 Framwork Programme)在ICT(Information & Communication Technologies)资金资助下的一个项目。为了推动流密码的发展,帮助密码研究人员开展对流密码的研究和设计,ECRYPT于2004年启动了e-STREAM流密码研究计划[1]。

LEX(Leak EXtraction)算法[2]是进入eSTREAM计划第二阶段的流密码候选算法,设计思想是基于分组密码算法AES[3,4]来构造流密码。与其它类似方式构造的流密码相比较,LEX的突出优点在于提高了密钥流的产生速度,使得加密效率约是AES算法的2.5倍[5]。

2 LEX算法

2.1 AES的基本运算

AES有4种基本运算,分别是字节代换(SB)、行移位(SR)、列混合(MC)和轮密钥加(ARK)。这4种运算是AES算法的核心操作和安全保证。

为便于下文分析论证LEX,此处列出列混合运算MC的相关等式。MC是状态a中的字节基于GF(2^8)域的变换,从而对每列数据起到良好的混淆作用[4]。第j列的变换关系为:

$$\begin{cases} a'_{0,j}=(\{02\}\cdot a_{0,j})+(\{03\}\cdot a_{1,j})+(\{01\}\cdot a_{2,j})+(\{01\}\cdot a_{3,j}) \\ a'_{1,j}=(\{01\}\cdot a_{0,j})+(\{02\}\cdot a_{1,j})+(\{03\}\cdot a_{2,j})+(\{01\}\cdot a_{3,j}) \\ a'_{2,j}=(\{01\}\cdot a_{0,j})+(\{01\}\cdot a_{1,j})+(\{02\}\cdot a_{2,j})+(\{03\}\cdot a_{3,j}) \\ a'_{3,j}=(\{03\}\cdot a_{0,j})+(\{01\}\cdot a_{1,j})+(\{01\}\cdot a_{2,j})+(\{02\}\cdot a_{3,j}) \end{cases} \tag{1}$$

2.2 LEX算法描述

LEX算法采用了128比特密钥和明文分组的AES算法作为密钥流产生部件。图1为LEX算法的初始化和运行过程:给定一个初始向量IV作为输入,在加密密钥K的作用下,用一次标准AES对IV进行加密,得到密文$S=AES_K(IV)$。此后,每执行一次AES会有10轮轮函数迭代,每一轮迭代后所得的128比特中间状态将有4个字节(共32比特)被提取出来作为输出密钥流的一部分,10轮迭代最终构成320比特的密钥流输出。

AES部件的10轮加密中,LEX算法对每一轮加密得到的数据的提取方式有所不同。如图2所示,方框内的一个$b_{i,j}$

本文受国家自然科学基金项目(61271254)资助。

李佳雨(1985—),男,硕士生,主要研究方向为保密技术与管理,E-mail:1175513011@qq.com;**石 会**(1982—),女,硕士,讲师,主要研究方向为保密技术与管理;**邓元庆**(1958—),男,硕士,教授,主要研究方向为保密技术与管理;**龚 晶**(1968—),女,硕士,副教授,主要研究方向为信息安全;**关 宇**(1962—),男,硕士,教授,主要研究方向为信息安全。

表示一个字节，第奇数轮提取 $b_{0,0}$，$b_{0,2}$，$b_{2,0}$，$b_{2,2}$ 4 个字节，第偶数轮提取 $b_{0,1}$，$b_{0,3}$，$b_{2,1}$，$b_{2,3}$ 4 个字节。

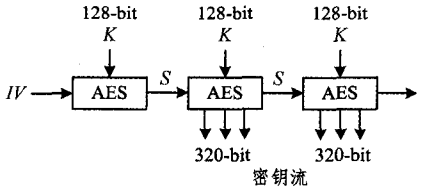

图 1　LEX 的初始化和密钥流生成

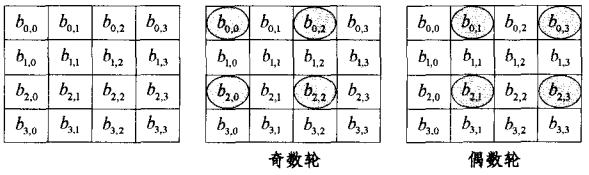

图 2　奇数轮和偶数轮提取字节位置

3　针对 LEX 的差分故障攻击

Boneh 等人利用密码计算过程中的错误来攻击基于 RSA-CRT 实现的签名方案，提出了故障攻击[6]，该攻击方法类属于侧信道攻击。随后，Biham 等人将故障攻击应用于私钥密码分析，提出了差分故障攻击的概念[7]。2010 年，Jiangyong Huang 等人提出了针对 LEX 的差分故障攻击方法[8]。

3.1　故障模型假设

在文献[8]中，作者针对 LEX-128 提出了诱导故障模型的基本假设：

(1)在 LEX 产生密钥流的过程中，攻击者能够针对其运行过程中的内部数据随机诱导 1 个比特的错误，并获取相对应的错误密钥流；

(2)攻击者能够重置被诱导错误的 LEX 部件，并对同一个中间状态多次诱导错误。

在该故障模型中，攻击者并不能确定诱导错误的准确位置，诱导的错误可能是 128 比特中间状态的任一位。在实际的实施过程中，可按照 Skorobogatov 验证成功的精确激光照射实现[9]。

3.2　故障位置确定方法

对 LEX 算法 AES 部件实施故障诱导，造成的 1 比特故障可能发生在 SB、SR、MC 或者 ARK 运算后的中间状态。根据故障位置的不同，可以将故障分为两大类。第一类故障是位于 MC 或 ARK 运算之后的数据，第二类故障是位于 SB 或 SR 运算之后的数据。

在故障模型的假设中，按照 Skorobogatov 提出的方法，虽然可以诱导中间状态 128 比特中某一位产生变化，但是并不能确定到底是 128 比特中的哪一位。因此，首先需要确定故障的位置。这一点可以采用观察比较诱导故障前后的密钥流差错来实现。

下面用图 3 来描述如何确定差错位置。假设该图开始的轮数 n 是奇数，则每轮提取的密钥流依次为 $\{g_{0,0}, g_{2,0}, g_{0,2}, g_{2,2}\}$、$\{l_{0,1}, g_{2,1}, g_{0,3}, g_{2,3}\}$ 和 $\{s_{0,0}, s_{2,0}, s_{0,2}, s_{2,2}\}$。

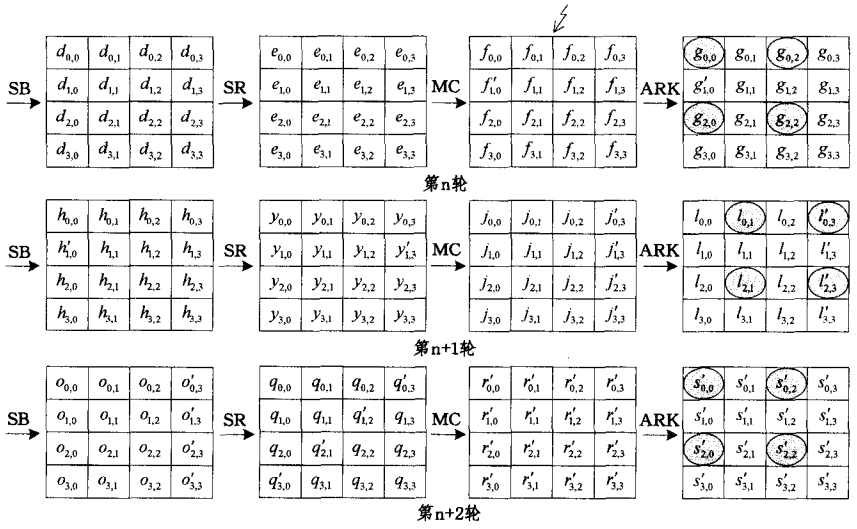

图 3　针对 LEX 的差分故障攻击

假设诱导故障的位置属于第一类，也就是处于 MC 或者 ARK 运算以后得到的状态中。而 MC 和 ARK 运算后的状态的故障位置，对后续各轮运算的影响是完全相同的。因此，可以进一步假设该故障发生在 MC 运算后的状态中，也就是某一个 $f_{i,j}$。为此，可以将 16 个 $f_{i,j}$ 分为 4 组，每组包含 4 个字节：

第一组：$f_{0,0}$，$f_{2,0}$，$f_{0,2}$，$f_{2,2}$；

第二组：$f_{1,0}$，$f_{2,1}$，$f_{3,2}$，$f_{0,3}$；

第三组：$f_{3,0}$，$f_{0,1}$，$f_{1,2}$，$f_{2,3}$；

第四组：$f_{1,1}$，$f_{3,1}$，$f_{1,3}$，$f_{3,3}$。

表 1 总结了故障位置和对后面两轮密钥流的影响。

表 1　故障位置的 4 个分组

(a)第一组

故障字节	第 n 轮	第 n+1 轮	第 n+2 轮
$f_{0,0}$	$g_{0,0}$	无	$s_{0,0}$，$s_{2,0}$，$s_{0,2}$，$s_{2,2}$
$f_{2,0}$	$g_{2,0}$	无	$s_{0,0}$，$s_{2,0}$，$s_{0,2}$，$s_{2,2}$
$f_{0,2}$	$g_{0,2}$	无	$s_{0,0}$，$s_{2,0}$，$s_{0,2}$，$s_{2,2}$
$f_{2,2}$	$g_{2,2}$	无	$s_{0,0}$，$s_{2,0}$，$s_{0,2}$，$s_{2,2}$

(b)第二组

故障字节	第 n 轮	第 n+1 轮	第 n+2 轮
$f_{1,0}$	无	$l_{0,3}$，$l_{2,3}$	$s_{0,0}$，$s_{2,0}$，$s_{0,2}$，$s_{2,2}$
$f_{2,1}$	无	$l_{0,3}$，$l_{2,3}$	$s_{0,0}$，$s_{2,0}$，$s_{0,2}$，$s_{2,2}$
$f_{3,2}$	无	$l_{0,3}$，$l_{2,3}$	$s_{0,0}$，$s_{2,0}$，$s_{0,2}$，$s_{2,2}$
$f_{0,3}$	无	$l_{0,3}$，$l_{2,3}$	$s_{0,0}$，$s_{2,0}$，$s_{0,2}$，$s_{2,2}$

(c)第三组

故障字节	第 n 轮	第 n+1 轮	第 n+2 轮
$f_{3,0}$	无	$l_{0,1}, l_{2,1}$	$s_{0,0}, s_{2,0}, s_{0,2}, s_{2,2}$
$f_{0,1}$	无	$l_{0,1}, l_{2,1}$	$s_{0,0}, s_{2,0}, s_{0,2}, s_{2,2}$
$f_{1,2}$	无	$l_{0,1}, l_{2,1}$	$s_{0,0}, s_{2,0}, s_{0,2}, s_{2,2}$
$f_{2,3}$	无	$l_{0,1}, l_{2,1}$	$s_{0,0}, s_{2,0}, s_{0,2}, s_{2,2}$

(d)第四组

故障字节	第 n 轮	第 n+1 轮	第 n+2 轮
$f_{1,1}$	无	无	$s_{0,0}, s_{2,0}, s_{0,2}, s_{2,2}$
$f_{3,1}$	无	无	$s_{0,0}, s_{2,0}, s_{0,2}, s_{2,2}$
$f_{1,3}$	无	无	$s_{0,0}, s_{2,0}, s_{0,2}, s_{2,2}$
$f_{3,3}$	无	无	$s_{0,0}, s_{2,0}, s_{0,2}, s_{2,2}$

从表中可以看出，对第 n 轮 MC 运算后的数据诱导故障后，通过观察第 n 轮和第 $n+1$ 轮密钥流的差错情况，可以将故障定位到某一个分组中。这也是如此分组的根据。

3.3 恢复第 $n+2$ 轮中 4 个字节的轮密钥

通过诱导 8 个故障，可以恢复出 8 字节的 $q_{i,j}$。通过这 8 字节的 $q_{i,j}$ 可以进一步恢复 4 字节第 $n+2$ 轮的轮密钥。

3.3.1 恢复 $q_{3,0}$ 和 $q_{1,2}$

如图 3 所示，具体操作步骤如下。

(1)求 $\Delta l_{1,3}$ 和 $\Delta l_{3,3}$。

通过对比诱导故障前后提取的密钥流，得到 $\Delta l_{0,3}$ 和 $\Delta l_{2,3}$ 的值。又因为：

$$l_{i,j}=ARK(j_{i,j}) \tag{2}$$

从而有：

$$\begin{cases}\Delta j_{0,3}=\Delta l_{0,3}\\ \Delta j_{2,3}=\Delta l_{2,3}\end{cases} \tag{3}$$

因而可以进一步得到 $\Delta j_{0,3}$ 和 $\Delta j_{2,3}$ 的值。

对于第 $n+1$ 轮的 MC 运算，根据式(1)，有：

$$\begin{cases}\Delta j_{0,3}=(\{02\}\cdot\Delta y_{0,3})+(\{03\}\cdot\Delta y_{1,3})+(\{01\}\cdot\Delta y_{2,3})+(\{01\}\cdot\Delta y_{3,3})\\ \Delta j_{1,3}=(\{01\}\cdot\Delta y_{0,3})+(\{02\}\cdot\Delta y_{1,3})+(\{03\}\cdot\Delta y_{2,3})+(\{01\}\cdot\Delta y_{3,3})\\ \Delta j_{2,3}=(\{01\}\cdot\Delta y_{0,3})+(\{01\}\cdot\Delta y_{1,3})+(\{02\}\cdot\Delta y_{2,3})+(\{03\}\cdot\Delta y_{3,3})\\ \Delta j_{3,3}=(\{03\}\cdot\Delta y_{0,3})+(\{01\}\cdot\Delta y_{1,3})+(\{01\}\cdot\Delta y_{2,3})+(\{02\}\cdot\Delta y_{3,3})\end{cases} \tag{4}$$

成立。式(4)中，$\Delta j_{0,3}$ 和 $\Delta j_{2,3}$ 通过式(3)已经求得，且 $\{\Delta y_{0,3}, \Delta y_{1,3}, \Delta y_{2,3}, \Delta y_{3,3}\}$ 中有 3 个元素为 0，则通过式(4)易得 $\Delta j_{1,3}$ 和 $\Delta j_{3,3}$ 的值。

由式(2)可知：

$$\begin{cases}\Delta l_{1,3}=\Delta j_{1,3}\\ \Delta l_{3,3}=\Delta j_{3,3}\end{cases} \tag{5}$$

成立。从而得到 $\Delta l_{1,3}$ 和 $\Delta l_{3,3}$ 的值。

(2)求 $\Delta o_{1,3}$ 和 $\Delta o_{3,3}$。

通过对比诱导故障前后提取的密钥流，得到 $\Delta s_{0,0}$ 和 $\Delta s_{2,0}$ 的值。因为：

$$s_{i,j}=ARK(r_{i,j}) \tag{6}$$

所以：

$$\begin{cases}\Delta r_{0,0}=\Delta s_{0,0}\\ \Delta r_{2,0}=\Delta s_{2,0}\end{cases} \tag{7}$$

成立。因此得到 $\Delta r_{0,0}$ 和 $\Delta r_{2,0}$。

对于第 $n+2$ 轮的 MC 运算，根据式(1)，有：

$$\begin{cases}\Delta r_{0,0}=(\{02\}\cdot\Delta q_{0,0})+(\{03\}\cdot\Delta q_{1,0})+(\{01\}\cdot\Delta q_{2,0})+(\{01\}\cdot\Delta q_{3,0})\\ \Delta r_{1,0}=(\{01\}\cdot\Delta q_{0,0})+(\{02\}\cdot\Delta q_{1,0})+(\{03\}\cdot\Delta q_{2,0})+(\{01\}\cdot\Delta q_{3,0})\\ \Delta r_{2,0}=(\{01\}\cdot\Delta q_{0,0})+(\{01\}\cdot\Delta q_{1,0})+(\{02\}\cdot\Delta q_{2,0})+(\{03\}\cdot\Delta q_{3,0})\\ \Delta r_{3,0}=(\{03\}\cdot\Delta q_{0,0})+(\{01\}\cdot\Delta q_{1,0})+(\{01\}\cdot\Delta q_{2,0})+(\{02\}\cdot\Delta q_{3,0})\end{cases} \tag{8}$$

成立。式(8)中共有 8 个变量，其中 $\Delta r_{0,0}$ 和 $\Delta r_{2,0}$ 已通过式(7)求得，且 $\Delta q_{0,0}=\Delta q_{1,0}=\Delta q_{2,0}=0$。那么，通过式(8)可以推导得到 $\Delta r_{1,0}$、$\Delta r_{3,0}$ 和 $\Delta q_{3,0}$。

同理可得 $\Delta r_{0,2}$、$\Delta r_{2,2}$、$\Delta r_{1,2}$、$\Delta r_{3,2}$ 和 $\Delta q_{1,2}$ 的值。

又因为：

$$q_{i,j}=SR(o_{i,j}) \tag{9}$$

所以：

$$\begin{cases}\Delta o_{13}=\Delta q_{12}\\ \Delta o_{33}=\Delta q_{33}\end{cases} \tag{10}$$

根据式(10)可得 $\Delta o_{1,3}$ 和 $\Delta o_{3,3}$ 的值。

(3)求 $q_{3,0}$、$q'_{3,0}$ 和 $q_{1,2}$、$q'_{1,2}$。

由于：

$$o_{i,j}=SB(l_{i,j}) \tag{11}$$

且通过前面的步骤已求得 $\Delta o_{1,3}$、$\Delta o_{3,3}$ 和 $\Delta l_{1,3}$、$\Delta l_{3,3}$，则通过查输入输出差分对数表，可以得到 $o_{1,3}$、$o'_{1,3}$ 和 $o_{3,3}$、$o'_{3,3}$[10]。但此时还无法将 $o_{1,3}$ 与 $o'_{1,3}$、$o_{3,3}$ 与 $o'_{3,3}$ 区分开。为此，还需要再一次诱导故障，重复上述步骤，得到 $o_{1,3}$、$o''_{1,3}$ 和 $o_{3,3}$、$o''_{3,3}$。将两次诱导故障推导得到的数据进行对比分析，可以得到准确的 $o_{1,3}$、$o'_{1,3}$、$o_{3,3}$ 和 $o'_{3,3}$。再根据式(9)，可以进一步恢复出 $q_{3,0}$、$q'_{3,0}$ 和 $q_{1,2}$、$q'_{1,2}$。

至此，$q_{3,0}$ 和 $q_{1,2}$ 求解完毕。

3.3.2 恢复其它 10 个字节的 $q_{i,j}$

参照以上步骤，通过诱导 $f_{0,0}$ 或者 $f_{2,2}$ 两次故障，可以求得 $q_{0,0}$ 和 $q_{2,2}$；通过诱导 $f_{0,1}$、$f_{1,2}$、$f_{2,3}$ 或者 $f_{3,0}$ 两次故障，可以求得 $q_{1,0}$ 和 $q_{3,2}$；通过诱导 $f_{0,2}$ 或者 $f_{2,0}$ 两次故障，可以求得 $q_{2,0}$ 和 $q_{0,2}$。

综上，通过诱导 8 次故障，可以恢复 8 字节 $q_{i,j}$，如图 4 左边第一个框所示。

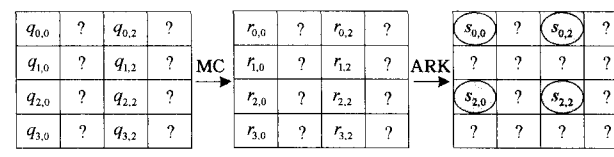

图 4 恢复 4 个轮密钥字节

3.3.3 恢复第 $n+2$ 轮中 4 个字节的轮密钥

观察图 4，$s_{i,j}(i,j\in\{0,2\})$ 可通过本轮提取的密钥流获取。根据方程组：

$$\begin{cases}s_{i,j}=r_{i,j}\oplus K_{i,j}^{n+2}, & i,j\in\{0,2\}\\ r_{i,j}=MC(q_{i,j}), & i\in\{1,2,3,4\}, j\in\{0,2\}\end{cases} \tag{12}$$

可以恢复出第 $n+2$ 轮的 4 个字节轮密钥 $K_{0,0}^{n+2}$、$K_{2,0}^{n+2}$、$K_{0,2}^{n+2}$、$K_{2,2}^{n+2}$。如图 5 中第 4 个框所示。

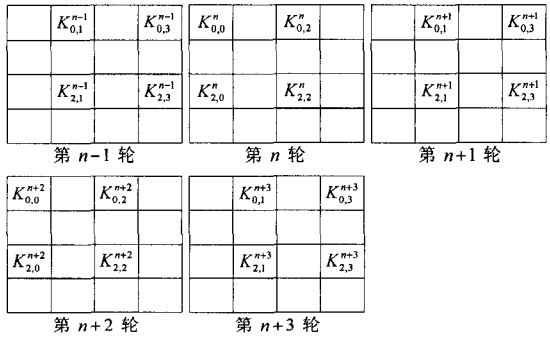

图 5　恢复的轮密钥

3.3.4　恢复第 $n-1$、n、$n+1$、$n+3$ 轮中 4 字节轮密钥

按照第 $n+2$ 轮提取密钥流的方法，可以每轮诱导 8 个故障，恢复第 $n-1$、n、$n+1$、$n+3$ 轮中的 4 个字节的轮密钥。总共 5 轮，诱导 40 个故障。每轮恢复的字节如图 5 所示。

需要注意的是，前面假设的第 n 轮是奇数轮，则第 $n-1$、$n+1$、$n+3$ 就是偶数轮。对于诱导故障的位置在 MC 或者 ARK 运算后的中间状态这类情况，偶数轮的诱导故障位置分组与奇数轮有所不同。以第 $n+1$ 轮为例，偶数轮的分组为 $\{j_{0,1}, j_{0,3}, j_{2,1}, j_{2,3}\}$、$\{j_{0,0}, j_{1,1}, j_{2,2}, j_{3,3}\}$、$\{j_{0,2}, j_{1,3}, j_{2,0}, j_{3,1}\}$、$\{j_{1,0}, j_{1,2}, j_{3,0}, j_{3,2}\}$。

3.3.5　恢复第 $n+2$ 轮中另外 10 字节的轮密钥

根据 AES 的轮密钥扩展算法和图 5 所示已经恢复出来的轮密钥，可以进一步恢复第 $n+2$ 轮中的另外 10 个字节的轮密钥，如图 6 所示。

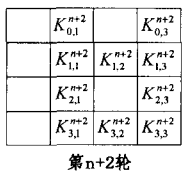

图 6　10 字节轮密钥

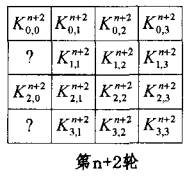

图 7　14 字节轮密钥

至此，总共恢复出第 $n+2$ 轮的 14 个字节的轮密钥（见图 7）。另外 2 字节可以用穷举法验证，需要 216 次计算。

综上所述，针对 LEX 算法的差分故障攻击，只需要 40 次诱导故障，216 次计算，就可以恢复出一轮 AES 加密的轮密钥。根据 AES 轮密钥扩展算法，连续的 4 列轮密钥就可以恢复出 AES 的加密密钥，从而破解 LEX。

4　抗差分故障攻击的改进与分析

从上述分析可以看出，LEX 抗差分故障攻击的能力不足，存在安全隐患，有必要对其进行改进。

4.1　抗差分故障攻击的 LEX 改进方法

文献[11]的作者在结语中建议，可以通过改变每轮密钥流字节的提取位置来抵抗差分故障攻击。Matt Henricksen 在文献[5]中也提出以与文献[11]相似的方式来增强 LEX 的安全性。然而，通过改变密钥流字节的提取位置来增强 LEX 的安全性是不可取的。其因有二：1）王秋艳等在文献[12]中分析指出，LEX 提取密钥流的方式是很安全的；2）通过 C++编程模拟的大量实验表明，改变提取密钥流的位置，将极大影响密钥流的随机性。特别是对于《随机性检测规范》中的单比特频数检测、扑克检测、重叠子序列检测、游程总数检测、游程分布检测等 5 种检测方法，1000 个 1000000 比特数据样本中，不能通过检测的样本个数的比例分别多达 17%、19%、6.9%、7.2%和 5.8%，远远大于《随机性检测规范》要求的 1.9%以内。

LEX 无法抵抗差分故障攻击，最根本的原因在于其提取密钥流的方式存在缺陷。通过将 LEX 每一轮提取的密钥流与诱导故障后的密钥流进行差错对比，很容易恢复出 AES 部件的种子密钥，从而破解 LEX 算法本身。

结合以上原因，在不改变 LEX 提取密钥流位置的前提下，对 LEX 进行了改进，不妨称改进的算法为 DLEX。

DLEX 保持 LEX 初始化模块不变，并按照 LEX 提取密钥流的方法，提取第二个 AES 部件的第 1、3、5、7 轮 SB 运算后数据相应位置共 16 字节作为一个随机序列，与后面每个 AES 部件的每一轮轮密钥先异或，然后将异或的结果作为该轮的轮密钥。以第 $n+2$ 轮为例，用 $m_{i,j}$ 代表随机序列的字节，用 $K_{i,j}^{n+2}$ 代表该轮的轮密钥，则该轮的操作步骤如图 8 所示。

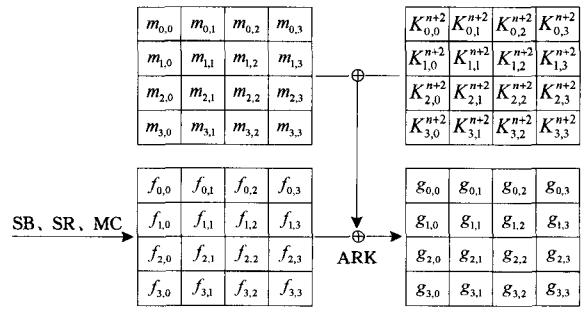

图 8　DLEX 算法

4.2　DLEX 算法抗差分故障能力分析

因为在差分故障攻击的过程中，都仅仅是用诱导故障前后密钥流的差值来恢复这些数据，所以按照针对 LEX 的差分故障攻击方法，可以恢复 DLEX 中如图 4 所示位置的 $q_{i,j}$ 等中间过程的值。

用 $mK_{i,j}^{n}$ 表示随机序列 $m_{i,j}$ 与第 n 轮的轮密钥 $K_{i,j}^{n}$ 按字节异或后的值，改进的 LEX 以 $mK_{i,j}^{n}$ 作为新的轮密钥，提高了 LEX 抗差分故障攻击的能力。证明分为以下 3 点。

定义 1　按照针对 LEX 算法的差分故障攻击方法，能且仅能恢复 DLEX 中与提取密钥流位置对应的轮密钥 $mK_{i,j}^{n}$。

证明：一方面，根据针对 LEX 算法的差分故障攻击方法，恢复轮密钥的过程中，是从诱导故障前后提取密钥流字节的差分入手，与密钥流字节的实际值无关。采用针对 LEX 算法的差分故障攻击方法对 DLEX 进行攻击，同样可以获取对 DLEX 诱导故障前后提取密钥流字节的差分，进而可以恢复与提取密钥流位置对应的轮密钥 $mK_{i,j}^{n}$，如图 9 所示。

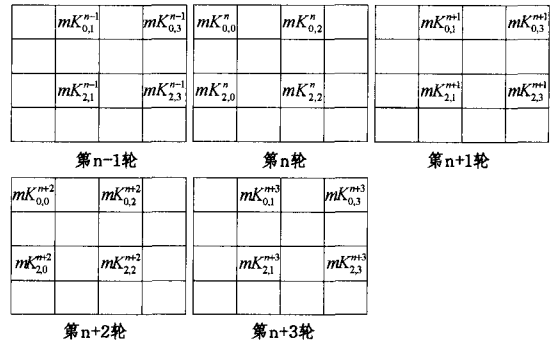

图 9　恢复 DLEX 的轮密钥

另一方面，由于恢复图 9 所示的轮密钥字节，是通过将 MC 后的数据与将提取的密钥流异或得到，因此仅能恢复提取密钥流位置处的轮密钥 $mK_{i,j}^{n}$。

定义 2 在 DLEX 中，根据图 9 已知的 $mK_{i,j}^{n}$，无法通过密钥扩展算法恢复更多的 $mK_{i,j}^{n}$。

证明：(反证法)假设可以通过密钥扩展算法求得 $mK_{i,j}^{n}$，则：

当 $j(\mathrm{mod}\ 4)\neq 0$ 时：

$$\begin{aligned} mK_{i,j}^{n+2} &= mK_{i,j}^{n+1} \oplus mK_{i,j-1}^{n+2} \\ &= (m_{i,j} \oplus K_{i,j}^{n+1}) \oplus (m_{i,j-1} \oplus K_{i,j-1}^{n+2}) \\ &= (m_{i,j} \oplus m_{i,j-1}) \oplus (K_{i,j}^{n+1} \oplus K_{i,j-1}^{n+2}) \\ &= (m_{i,j} \oplus m_{i,j-1}) \oplus K_{i,j}^{n+2} \end{aligned} \tag{13}$$

且

$$mK_{i,j}^{n+2} = m_{i,j} \oplus K_{i,j}^{n+2} \tag{14}$$

从而由式(13)和式(14)得：

$$m_{i,j-1} \equiv 0 \tag{15}$$

显然式(15)不恒成立，从而假设不成立，故而当 $j(\mathrm{mod}\ 4)\neq 0$ 时，无法通过密钥扩展算法求得更多的 $mK_{i,j}^{n}$。

当 $j(\mathrm{mod}\ 4)=0$，也即 $j=0$ 时，根据假设，有：

$$\begin{aligned} mK_{i,0}^{n+2} &= (mK_{i,0}^{n+1}) \oplus SB(SR(mK_{i,3}^{n+1})) \oplus Rcon(n+2) \\ &= (m_{i,0} \oplus K_{i,0}^{n+1}) \oplus SB(SR(m_{i,3} \oplus K_{i,3}^{n+1})) \oplus Rcon \\ &\quad (n+2) \end{aligned} \tag{16}$$

且

$$\begin{aligned} mK_{i,0}^{n+2} &= m_{i,0} \oplus K_{i,0}^{n+2} \\ &= m_{i,0} \oplus (K_{i,0}^{n+1} \oplus SB(SR(K_{i,3}^{n+1})) \oplus Rcon(n+2)) \end{aligned} \tag{17}$$

由式(16)和式(17)得：

$$m_{i,3} \equiv 0 \tag{18}$$

显然式(18)不恒成立，从而假设不成立。

因此，无法通过密钥扩展算法恢复更多的 $mK_{i,j}^{n}$。

定义 3 在 DLEX 中，根据图 9 已知的 $mK_{i,j}^{n}$，无法通过密钥扩展算法恢复更多的 $K_{i,j}^{n}$。

证明：当 $j(\mathrm{mod}\ 4)\neq 0$ 时：

$$\begin{aligned} K_{i,j}^{n+2} &= K_{i,j}^{n+1} \oplus K_{i,j-1}^{n+2} \\ &= (m_{i,j} \oplus mK_{i,j}^{n+1}) \oplus (m_{i,j-1} \oplus mK_{i,j-1}^{n+2}) \\ &= (m_{i,j} \oplus m_{i,j-1}) \oplus (mK_{i,j}^{n+1} \oplus mK_{i,j-1}^{n+2}) \end{aligned} \tag{19}$$

因为 $m_{i,j} \oplus m_{i,j-1}$ 是无法获取的，故通过密钥扩展算法无法得到 $K_{i,j}^{n+2}$。

当 $j(\mathrm{mod}\ 4)=0$，也即 $j=0$ 时：

$$\begin{aligned} K_{i,0}^{n+2} &= (K_{i,0}^{n+1}) \oplus SB(SR(K_{i,3}^{n+1})) \oplus Rcon(n+2) \\ &= (m_{i,0} \oplus mK_{i,0}^{n+1}) \oplus SB(SR(m_{i,3} \oplus mK_{i,3}^{n+1})) \oplus \\ &\quad Rcon(n+2) \\ &= (m_{i,0} \oplus SB(SR(m_{i,3}))) \oplus (mK_{i,0}^{n+1} \oplus SB(SR \\ &\quad (mK_{i,3}^{n+1})) \oplus Rcon(n+2)) \end{aligned} \tag{20}$$

因为 $m_{i,0}$ 和 $m_{i,3}$ 是无法获取的，故通过密钥扩展算法无法得到 $K_{i,j}^{n+2}$。

综合以上 3 条证明可以得知，采用差分故障攻击方法虽然可以恢复 DLEX 中与提取密钥流字节对应位置的轮密钥，但是无法通过轮密钥扩展算法获得更多的轮密钥，从而无法通过这种方法恢复 AES 部件的种子密钥。

所以，改进的 LEX 能够抵抗差分故障攻击。

4.3 速度分析

LEX 算法中，每组 128 比特的 AES 加密算法可以得到 320 比特的密钥流输出。因此，LEX 算法的速度约为 AES 的 2.5 倍(320/128≈2.5)。DLEX 算法仅仅是在每一个密钥流提取模块增加了一次 128 比特的异或运算，对整个算法的速度无实质性影响，几乎是完全继承了原算法的速度优越性。

4.4 密钥流随机性检测实例

密钥流的随机性是评判流密码算法优劣的一个重要指标。2012 年，国家密码管理局第 23 号公告批准了编号为 GM/T 0005-2012 的《随机性检测规范》[12]，该规范共收纳了 15 种随机性检测方法，它们分别是：单比特频数检测、块内频数检测、扑克检测、重叠子序列检测、游程总数检测、游程分布检测、块内最大 1 游程检测、二元推导检测、自相关检测、矩阵秩检测、累加和检测、近似熵检测、线性复杂度检测、Maurer 通用统计检测、离散傅里叶检测。该《随机性检测规范》针对随机数发生器的性能评价步骤如下：

(1)用随机数发生器产生 1000 个长度为 1000000 位的 0-1随机数序列作为待检样本。

(2)对产生的 1000 个样本，依次使用上述 15 种随机性检测方法进行检测。

(3)对于每一种随机性检测方法，统计 P-value 值不小于显著性水平 α(该规范建议取 $\alpha=0.01$)的样本个数。单个样本的 P-value 不小于 α，表示该样本通过该项检测。

(4)记样本数量为 s，则通过检测的样本个数应当不少于 $s(1-\alpha-3\sqrt{\alpha(1-\alpha)/s})$。当样本数量为 1000 时，如果通过的样本个数不少于 981(未通过的样本个数不多于 19)，则随机数发生器通过此项检测；否则未通过此项检测。

如果随机数发生器通过该规范规定的所有检测项目，则随机数发生器通过该规范检测；否则，未通过该规范检测。

本文采用 C++编程工具，分别编写了基于 LEX 和 DLEX 算法的密钥流产生器的程序，并依据《随机性检测规范》搭建了随机性检测平台，分别对 LEX 和 DLEX 产生的 1000 个 1000000 位 0-1 随机序列进行了随机性检测。为了避免由于密钥流产生器的输入不同而影响评价结果，每产生一个样本时，LEX 和 DLEX 都采用了相同的输入向量和密钥。

对 LEX 和 DLEX 各自产生的 1000 个样本分别进行 15 种随机性检测，统计数据见表 2。

表 2 样本随机性统计分析

检测方法	未通过检测的样本数		P-value 值的期望		P-value 值的方差	
	LEX	DLEX	LEX	DLEX	LEX	DLEX
单比特频数检测	6	10	0.502	0.494	0.082	0.086
块内频数检测	10	7	0.500	0.493	0.087	0.079
扑克检测	2	11	0.512	0.503	0.082	0.085
重叠子序列检测	9	8	0.492	0.511	0.082	0.081
游程总数检测	13	5	0.509	0.516	0.081	0.081
游程分布检测	13	15	0.478	0.486	0.083	0.077
块内最大 1 游程检测	10	15	0.493	0.496	0.088	0.080
二元推导检测	7	7	0.494	0.504	0.081	0.084
自相关检测	9	13	0.520	0.485	0.084	0.084
矩阵秩检测	18	17	0.507	0.502	0.082	0.079
累加和检测	7	11	0.502	0.499	0.081	0.084
近似熵检测	8	10	0.496	0.498	0.084	0.083
线性复杂度检测	14	6	0.493	0.498	0.082	0.084
Maurer 通用统计检测	14	16	0.482	0.482	0.083	0.084
离散傅里叶检测	0	0	0.502	0.509	0.083	0.082

(下转第 377 页)

文本水印技术，该技术在不对文本做任何改变的情况下实现水印信息的嵌入和提取，具有很好的隐蔽性，可以抵抗多种常见的针对文本水印的攻击。算法构造简单、易于实现，可据此设计出具有实用意义的文本水印嵌入与提取系统。

参考文献

[1] Atallah M, McDonough C, Nirenburg S, et al. Natural Language Processing for Information Assurance and Security: An Overview and Implementations[C]// Proceedings of the 9th ACM/SIGSAC New Security Paradigms Workshop. Ireland, 2000: 51-65

[2] 刘旻昊，孙堡垒，郭云彪，等. 文本数字水印技术研究综述[J]. 东南大学学报(自然科学版)，2007，37(z1)：225-230

[3] 黄华，齐春，李俊，等. 文本数字水印[J]. 中文信息学报，2001，15(5)：52-57

[4] 梁旭，远志永，黄明. 基于行间距编码的文本数字水印算法[J]. 信息技术，2008，32(3)：38-41

[5] 于晨斐. 基于二次余数的 Word 文档数字水印[J]. 计算机仿真，2007，24(11)：324-326

[6] 蔡菲菲，刘洋，尹香兰. 一种基于 word 文档的文本水印技术研究[J]. 计算机科学，2012，39(z3)：39-40

[7] 陈翔. 一种基于中文字符编码的文本水印算法研究[J]. 计算机技术与发展，2013(2)：237-240

[8] Maxemchuk N F. Electronic Document Distribution[J]. AT&T Bell Laboratories Technical Journal, 1994, 73(5): 73-80

[9] Atallah M J, Raskin V, Meta C. Natural language watermarking: design, analysis, and a proof-of-concept implementation[C]// Proc. of Information Hiding-Fourth International Workshop. Berlin: Springer, 2001: 185-200

[10] 张宇，刘挺，陈毅恒，等. 自然语言文本水印[J]. 中文信息学报，2005，19(1)：56-62

[11] 杨超，李仁发，蒋斌，等. 基于语义的自然语言文本数字水印研究[J]. 计算机工程与设计，2005，26(6)：1428-1430

[12] 甘灿，孙星明，刘玉玲，等. 一种改进的基于同义词替换的中文文本信息隐藏方法[J]. 东南大学学报，2007，37(z1)：137-140

[13] Jalil Z, Mirza A M. A Review of Digital Watermarking Techniques for Text Documents[C]// International Conference on Information and Multimedia Technology. Jeju Island, 2009: 230-234

[14] 温泉，孙锬锋，王树勋. 零水印的概念与应用[J]. 电子学报，2003，31(2)：214-216

[15] 斯琴，张力，廉德亮. 基于文本特征的文本水印算法[J]. 计算机应用，2009，29(9)：2348-2350

[16] 舒娟娟，刘玉玲. 基于词性频率的中文文本零水印算法[J]. 计算机应用，2011，31(z2)：103-105

[17] Che Wan-xiang, Li Zheng-hua, Liu Ting. LTP: A Chinese Language Technology Platform[C]// Proc. of the Coling 2010: Demonstrations. Beijing, 2010: 13-16

(上接第 356 页)

根据《随机性检测规范》的原理，对多个样本进行随机性检测，统计得到的 P-value 值应当均匀分布在闭区间[0,1]内，其均值约为 0.5。对表 1 进行数据分析可以看出，LEX 和 DLEX 两种算法产生样本的 P-value 值的期望值均在 0.5 左右的一定区间内分布，并且两者的 P-value 值的期望值和方差都非常接近，符合统计规律，说明了 DLEX 算法与 LEX 算法产生的随机序列其随机性的统计特性相当。

结束语 LEX 算法以分组密码 AES 作为密钥流生成的核心部件，架起了分组密码与流密码之间的桥梁。但是由于其按奇偶数轮固定位置提取 4 个字节轮密钥加 ARK 的结果作为密钥流，导致其对于差分故障攻击存在安全隐患。本文分析了针对 LEX 算法的差分故障攻击，采取将一个随机序列与每组轮密钥异或的结果作为新的轮密钥的方法对 LEX 进行了改进，并用一个实例仿真来验证了改进算法的密钥流随机性。结果表明，改进的 LEX 抗差分故障攻击的能力得到了增强，且具有与原算法相当并符合国家规范的密钥流随机性，增强了 LEX 算法的密码性能。

参考文献

[1] 刘依依. eSTREAM 和流密码分析现状[J]. 信息安全与通信保密，2009，31(12)：47-49

[2] Alex B. A new 128 bit key stream cipher LEX[EB/OL]. [2005-06-13]. http://www.ecrypt.eu.org/stream/ciphers/lex/lex.pdf

[3] National Institute of Standards and Technology (NIST). Announcing the Advanced Encryption Standard (AES) [EB/OL]. (2001-11-26). http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[4] 邓元庆，龚晶，石会. 密码学简明教程[M]. 北京：清华大学出版社，2011：71-93

[5] Henricksen M. Flexible Block Ciphers: Modifying LEX [C]// Proceedings of ICCSIT2010. Chengdu, 2010

[6] Boneh D, DeMillo R A, Lipton R J. On the importance of checking cryptographic protocols for faults[M]// Advances in Cryptology—EUROCRYPT'97. 1997: 37-51

[7] Biham E, Shamir A. Differential fault analysis of secret key cryptosystems[M]// Advances in Crytpology—CRYPTO'97. 1997: 37-51

[8] Huang J, Susilo W, Seberry J. Differential fault analysis of LEX [C]// Proceedings of SCN 2010. Amalfi, Italy, 2010

[9] Skorobogatov S, Anderson R. Optical fault induction attacks[M]// Cryptographic Hardware and Embedded System-CHES 2002. 2003: 2-12

[10] Giraud C. DFA on AES[M]// Advanced Encryption Stardard-AES. 2005: 27-41

[11] 张中亚，关杰. 对流密码算法 LEX 的差分故障攻击[J]. 上海交通大学学报，2012，40(6)：865-869

[12] 王秋燕，金晨辉. LEX 算法的输出位置分析[J]. 计算机应用与软件，2014，31(10)：309-313

[13] 随机性检测规范：GM/T 0005-2012[S]. 北京：中国标准出版社，2012