CLEFIA算法的不可能差分密码分析

殷广丽 卫宏儒
(北京科技大学数理学院 北京 100083)

摘 要 为研究分组密码CLEFIA抵抗不可能差分攻击的能力,使用了两类9轮不可能差分路径,给出了相关攻击结果。基于一条9轮不可能差分路径,利用轮函数中S盒差分分布表恢复密钥,攻击了11轮的CLEFIA。改进了关于14轮的CLEFIA-256的不可能差分攻击的结果,将数据复杂度降低到$2^{104.23}$,时间复杂度降低到$2^{221.5}$。同时,在两条不可能差分的基础上,根据轮密钥之间的关系,使用Early-abort技术和S盒差分分布表,分别给出12轮CLEFIA-128和13轮CLEFIA-128的不可能差分攻击。

关键词 分组密码,CLEFIA,不可能差分,S盒,Early-abort技术
中图法分类号 TN918.1 **文献标识码** A

Impossible Differential Cryptanalysis of CLEFIA

YIN Guang-li WEI Hong-ru
(School of Mathematics and Physics, University of Science and Technology Beijing, Beijing 100083, China)

Abstract To analyze impossible differential cryptanalysis on the block cipher CLEFIA, the results are presented based on two 9-round impossible differential role. It uses the output and input differences of S-boxes to recover round keys, which apply one impossible differential attack to 11-round reduced CLEFIA. Improved result on 14 round CLEFFIA-256 is given with the number of chosen plaintexts being reduced to $2^{104.23}$ and the time complexity reduced to $2^{221.5}$. At the same time, exploiting the key relations, using Early -Abort technique and S-boxes differential technique, impossible differential attack is proposed on 12 and 13-round CLEFIA-128 based on two impossible differential roles respectively.

Keywords Block cipher, CLEFIA, Impossible differential attack, S-box, Early-abort technique

1 前言

不可能差分分析是由Biham在文献[1]中提出来的,它作为差分密码分析的变形,以简单的分析过程和高效的攻击能力引起了人们的广泛关注。该密码分析方法不仅成功攻击了大量分组密码,如AES, ARIA, FOX, SMS4, Camellia, LBlock, CLEFIA等,同时也分析了一些常用的算法结构,如Feistel和Rijndael结构等。不可能差分分析一般包括两个步骤:构造不可能差分路径和恢复密钥。不可能差分路径的构造是通过寻找加密方向概率为1的差分路径和解密方向概率为1的差分路径,来寻找这两条差分路径在中间处所构成的矛盾,从而得到一条概率为0的路径。在恢复密钥阶段,利用构造的不可能差分链,得到满足差分链前后若干轮的明密文对,猜测相应的轮密钥,部分加解密后,所有符合该不可能差分链的候选密钥都是错误的,通过筛选掉这些错误的密钥猜测,再结合其它技术如穷举搜索等可恢复正确密钥。

分组密码算法CLEFIA[2,3]是由索尼公司的Shirai等在FSE2007上公布的。CLEFIA的分组长度为128比特,密钥长度为128/192/256,分别记为CLEFIA-128, CLEFIA-192, CLEFIA-256。CLEFIA安全性分析报告[2]中给出了两条9轮的不可能差分路径,王薇等人[4]基于一个不可能差分路径,提出生日筛选算法,将CLEFIA的不可能差分攻击扩展到12轮的CLEFIA-128/192/256,13轮CLEFIA-192/256以及14轮的CLEFIA-256,改进了不可能差分的攻击结果。Tsunoo[5]等人结合线性变换的分支数为5,提出了新的9轮的不可能差分路径,并将攻击结果扩展到12轮的CLEFIA-128/192/256、13轮的CLEFIA-192/256以及14轮的CLEFIA-256,进一步改进了对CLEFIA的不可能差分攻击结果。孙兵等人[6]通过研究线性变换,也给出新的9轮不可能差分路径,并基于此路径成功攻击了12轮的CLEFIA-128、13轮的CLEFIA-192、14轮的CLEFIA-256。唐学海和Hamid Mala等人[7,8]都使用了两类9轮不可能差分路径,结合CLEFIA-128轮子密钥之间的关系,并使用Early-Abort技术成功攻击了带白化与不带白化的13轮CLEFIA-128。

本文基于文献[6]提出的9轮不可能差分路径,利用S盒差分分布表攻击11轮CLEFIA,在恢复相同密钥长度的情况下,计算复杂度降低了2^8,同时改进了文献[6]中14轮CLEFIA-256的攻击结果,大大降低了计算复杂度和数据复杂度。

本文受2013年国家自然科学基金(61272476)和内蒙古自治区科技创新引导奖励资金(2012)资助。
殷广丽(1989—),女,硕士生,主要研究方向为密码学,E-mail:374691744@qq.com;卫宏儒(1963—),男,硕士,副教授,主要研究方向为数学、信息安全与密码学、物联网关键技术。

利用轮子密钥之间的关系，结合S盒差分分布表方法，对12轮CLEFIA-128和13轮CLEFIA-128进行了攻击。本文第2节介绍CLEFIA；第3节列举了关于CLEEFIA的一些理论；第4节用不可能差分分析CLEFIA算法。

2 CLEFIA算法

分组密码CLEFIA分组长度为128bit，CLEFIA-128/192/256对应轮数r分别为18、22、26。加密中用了个32bit子密钥$(RK_0,RK_1,\cdots,RK_{2r-1})$和4个32比特的密钥$(WK_0,WK_1,WK_2,WK_3)$。图1示出CLEFIA的加密结构。

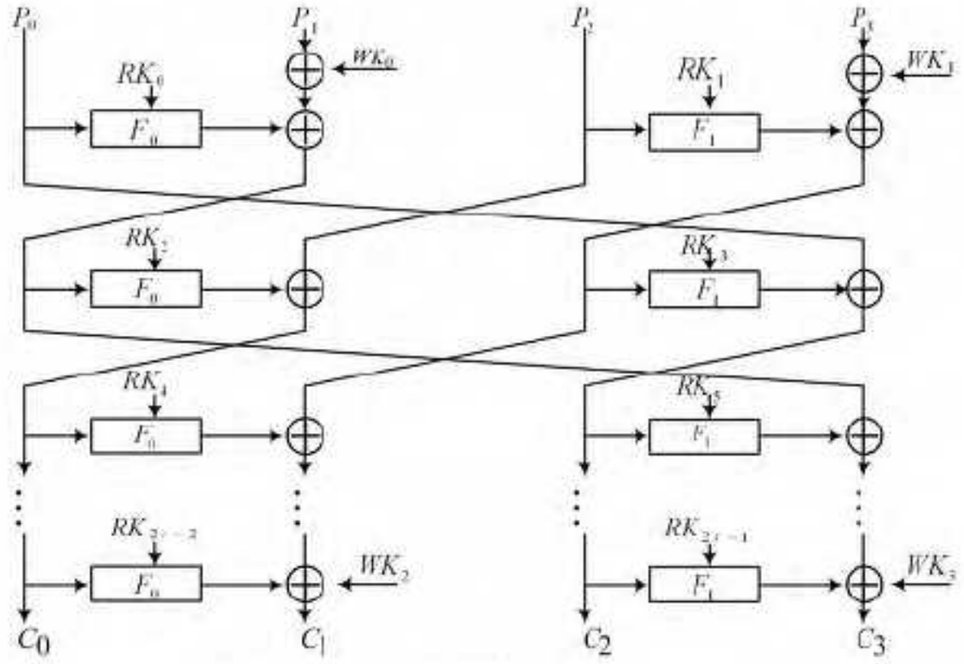

图1 CLEFIA加密结构示意图

2.1 符号说明

$P=P_0|P_1|P_2|P_3$：表示128比特输入，每个$P_i(i=0,1,2,3)$为32比特；

$C=C_0|C_1|C_2|C_3$：表示128比特输出，每个$C_i(i=0,1,2,3)$为32比特；

$\Delta P,\Delta C$：分别为明文差分和密文差分；

$RK_{i,j}$：轮子密钥RK_i的第j个字节；

$RK_{i,(j,k)}$：轮子密钥RK_i的第j和k个字节；

$RK_i[j-k]$：轮子密钥RK_i的第j到k比特；

$M_i(a)$：矩阵M_i对输入32比特字a做运算。

2.2 CLEFIA算法介绍[9]

(1)加密过程

1)将128比特明文分成4个32比特的字$P=(P_0,P_1,P_2,P_3)$，明文与白化子密钥异或，即$(X_0^0,X_1^0,X_2^0,X_3^0)=(P_0,P_1\oplus WK_0,P_2,P_3\oplus WK_1)$，密文记为$C=(C_0,C_1,C_2,C_3)$。

2)对$i=1,\cdots,r$

$X_0^i=X_1^{i-1}\oplus F_0(X_0^{i-1},RK_{2i-2});X_1^i=X_2^{i-1}$

$X_2^i=X_3^{i-1}\oplus F_1(X_2^{i-1},RK_{2i-1});X_3^i=X_0^{i-1}$

最后一轮进行输出变换

$X_0^r=X_0^{r-1};X_1^r=X_1^{r-1}\oplus F_0(X_0^{r-1},RK_{2r-2})$

$X_2^r=X_2^{r-1};X_3^r=X_3^{r-1}\oplus F_1(X_2^{r-1},RK_{2r-1})$

3)将128比特的$(X_0^r,X_1^r,X_2^r,X_3^r)$与两个白化子密钥异或，得到密文即$C=(C_0,C_1,C_2,C_3)$。其中$F_0$和$F_1$都用到了两个不同的$8\times8$的S盒$S_0$和$S_1$，$F_0$是将32比特的输入$x$和32比特的子密钥$k$分别划分成4个字节，即$x=(x_0,x_1,x_2,x_3),k=(k_0,k_1,k_2,k_3)$。先逐比特异或，后进入S盒即$S(x\oplus k)=(S_0(x_0\oplus k_0),S_1(x_1\oplus k_1),S_0(x_2\oplus k_2),S_1(x_3\oplus k_3))$。最后一步是经过由矩阵$M_0$确定的扩散变换，定义如下：

$$(v_0,v_1,v_2,v_3)^{\mathrm{T}}=M_i\cdot(u_0,u_1,u_2,u_3)^{\mathrm{T}}(i=0,1)$$

$$M_0=\begin{pmatrix}01&02&04&06\\02&01&06&04\\04&06&01&02\\06&04&02&01\end{pmatrix}$$

$$M_1=\begin{pmatrix}01&08&02&0a\\08&01&0a&02\\02&0a&01&08\\0a&02&08&01\end{pmatrix}$$

轮函数F是将S_0和S_1调换位置，M_0由M_1代替的函数。

(2)r轮CLEFIA-128密钥扩展

定义1 $\sum:\{0,1\}^{128}\rightarrow\{0,1\}^{128}$函数为

$X_{(128)}\rightarrow Y_{(128)}$

$Y=X[7-63]|X[121-127]|X[0-6]|X[64-120]$

CLEFIA-128的密钥扩展算法分为两部分：由种子密钥K生成L；由K和L扩展得到白化子密钥$WK_i(0\leqslant i\leqslant3)$和$RK_j(0\leqslant j<2r)$。$GFN_{4,12}$表示一类广义Feistel结构，它包含$d$个32比特分支和12轮迭代，在此不详写。子密钥$CON_i^{128}$$(0\leqslant i<24)$为24个32比特的常数。

下面为具体密钥扩展算法：

第1步 $L\leftarrow GFN_{4,12}(CON_0^{128},\cdots,CON_{23}^{128},K_0,\cdots,K_3)$；

第2步 $WK_0|WK_1|WK_2|WK_3\leftarrow K$；

第3步 For $i=0$ to $i=8$

$\{T=L\oplus(CON_{24+4i}^{128}|CON_{24+4i+1}^{128}|CON_{24+4i+2}^{128}|CON_{24+4i+3}^{128})$

$L\leftarrow\sum(L)$

当i是奇数时：$T=T\oplus K$；

$RK_{4i}|RK_{4i+1}|RK_{4i+2}|RK_{4i+3}\leftarrow T\}$

3 CLEFIA的几点理论

定理1[4] 对F函数(F_0,F_1)，若已知两个32比特的输入(In,In')及相应的输出差分ΔOut，则求解F函数的子密钥RK只需一次F运算。

定理2[4] 对简化为r轮的CLEFIA分组密码算法，令(RK_{2r-3},RK_{2r-4})为第$r-1$轮的子密钥，(RK_{2r-1},RK_{2r-2})为第r轮的子密钥，(WK_2,WK_3)为最后一轮的白化子密钥，$C^r=(C_0^r,C_1^r,C_2^r,C_3^r)$为密文，则子密钥$RK_{2r-3},RK_{2r-4}$和白化子密钥$WK_2,WK_3$的关系为

$$WK_3\oplus RK_{2r-4}=InS_{F_0}^{r-1}\oplus F_1^r(C_2^r,RK_{2r-1})\oplus C_3^r$$

$$WK_2\oplus RK_{2r-3}=InS_{F_1}^{r-1}\oplus F_0^r(C_0^r,RK_{2r-2})\oplus C_1^r$$

其中，$InS_{F_0}^{r-1}$和$InS_{F_1}^{r-1}$分别表示第$r-1$轮的F_0^{r-1}和F_1^{r-1}中S盒的输入。另外，$WK_0\oplus RK_2$和$WK_1\oplus RK_3$存在线性表达式，如下：

$$WK_0\oplus RK_2=InS_{F_0}^2\oplus F_0^1(P_0,RK_0)\oplus P_1$$

$$WK_1\oplus RK_3=InS_{F_1}^2\oplus F_1^1(P_2,RK_1)\oplus P_3$$

性质1[7] 若已知32比特RK_{24}，可以计算10比特的$RK_3[18-20,25-31]$，22比特$RK_1[10-31]$。

性质2[7] 若已知32比特RK_{25}，可以计算10比特$RK_2[22-31]$，22比特$RK_3[0-17,21-24]$。

性质3[5] 9轮不可能差分路径：对9轮的CLEFIA算法，若明文差分为$(0,000\alpha,0,0)$，则输出差分不可能为$(0,\beta000,0,0)$，这条不可能差分路径记为$(0,000\alpha,0,0)\xrightarrow{9}(0,\beta000,0,0)$。

性质 4[6] 9 轮不可能差分路径：对 9 轮的 CLEFIA 算法，若明文差分为 $(0,00\alpha\beta,0,0)$，则输出差分不可能为 $(0,00\gamma0,0,0)$，这条不可能差分路径记为 $(0,00\alpha\beta,0,0)\xrightarrow{9}(0,00\gamma0,0,0)$。

4 CLEFIA 算法的不可能差分攻击

本节主要研究了用两类不可能差分路径分别分析 CLEFIA。应用性质 4 中的 9 轮不可能差分路径对 11 轮的 CLEFIA、12 轮的 CLEFIA-128 和 14 轮 CLEFIA-256 进行了攻击。应用性质 3 中的 9 轮不可能差分路径，给出了对 13 轮的 CLEFIA-128 带白化子密钥的不可能差分攻击。

4.1 11 轮 CLEFIA 的不可能差分攻击

4.1.1 11 轮的不可能差分(1)

研究在 9 轮不可能差分路径 $(0,00\alpha\beta,0,0)\xrightarrow{9}(0,00\gamma0,0,0)$ 末尾加两轮的情况，攻击 2—12 轮。根据定理 2 可恢复 72 比特密钥 $(RK_{18}\oplus WK_3)_2, RK_{20}, RK_{21}$。具体攻击过程如下：

(1)明密文对的选取

选取 2^N 个明文结构，每个结构中明文具有如下形式：$P_0, P_1\oplus 00\alpha\beta, P_2, P_3$，其中 P_0, P_1, P_2, P_3 为固定 32 比特字，则每个结构中有 2^{16} 个元素，生成 2^{31} 个明文对，2^N 个结构共形成 2^{N+31} 个明文对。对这些明文对加密 11 轮，得到相应的密文对，保留满足差分形式为 $C\oplus C^*=(M_0(00*0),****,00\gamma0)$ 的密文对，剩余明密文对个数为 $2^{N+31}\times 2^{-80}=2^{N-49}$。

(2)密钥的恢复

对选取的明密文对，猜测 RK_{21}，根据 S 盒差分分布表，恢复 $(RK_{18}\oplus WK_3)_2, RK_{20}$，对每个明密文对，$(RK_{18}\oplus WK_3)_2, RK_{20}, RK_{21}$ 这 72 比特密钥剩余的概率为 2^{-40}，经过 2^{N-49} 个明密文对，若 $N=94.7$，剩余错误密钥个数为

$$2^{72}\times(1-2^{-40})^{2^{N-49}}<1$$

复杂度分析：

1)选取明密文对需要 $2^{110.7}$ 次加密；

2)在密钥恢复阶段，由于要猜测 RK_{21}，因此至少需要 $2^{32}\times 2^{45.7}=2^{77.7}$ 次 F 函数，约为 2^{73} 次加密。

因此，该攻击的数据复杂度为 $2^{110.7}$，恢复密钥时间复杂度为 2^{73} 次加密运算。

4.1.2 11 轮的不可能差分(2)

再研究利用 9 轮不可能差分路径 $(0,00\alpha\beta,0,0)\xrightarrow{9}(0,00\gamma0,0,0)$，在前后各加 1 轮，对 11 轮的 CLEFIA 进行攻击。

(1)选择明文对

选择 2^N 个结构，每个结构满足如下明文差分：

$$P\oplus P^*=(0,0,00\alpha\beta,M_1(00**))$$

则 2^N 个结构共形成 2^{N+63} 个明文对。筛选满足密文差分为 $(00\gamma0,M_0(00*0),0,0)$ 的密文对，剩余密文对 $2^{N+63}\times 2^{-112}=2^{N-49}$ 个。

(2)对上述的密文对 (C,C^*) 及对应的明文对，恢复 $RK_{1,(2,3)}$ 和 $RK_{20,2}$ 共 3 个字节，剩余密钥的概率为 2^{-24}，取 $N=77.06$，剩余错误密钥的个数为

$$2^{24}\times(1-2^{-24})^{2^{N-49}}<1$$

则数据复杂度为 $2^{109.06}$。时间复杂度为 2^{24} 次加密运算。

4.2 12 轮 CLEFIA-128 的不可能差分攻击

分析利用性质 4 中的 9 轮不可能差分路径 $(0,00\alpha\beta,0,0)\xrightarrow{9}(0,00\gamma0,0,0)$ 前加 1 轮，后加 2 轮，攻击第 2—13 轮的 CLEFIA-128 如图 2 所示，由性质 1 和性质 2 知，恢复 RK_{25}，$RK_{24}, RK_{3,(2,3)}, (RK_{22}\oplus WK_3)_2$ 共 72 比特密钥。攻击过程如下。

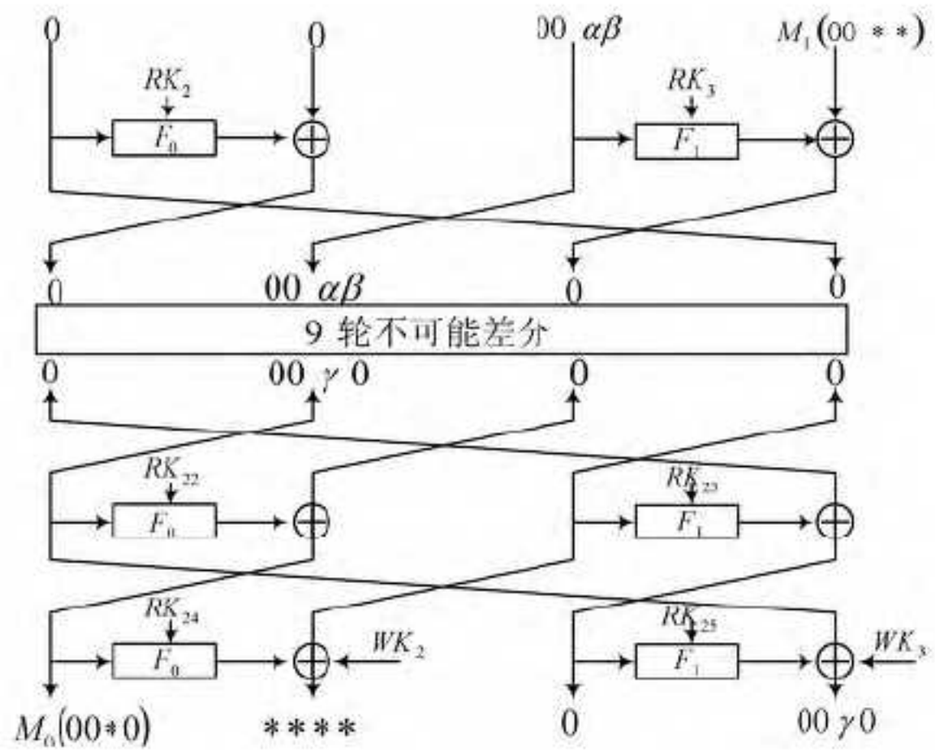

图 2 对 12 轮 CLEEIA-128 的不可能差分攻击

(1)选择明文结构

令每个结构中的明文对满足差分形式为

$$P\oplus P^*=(0,0,00\alpha\beta,M_1(00**))$$

该明文结构包括 2^{32} 个明文，可形成 2^{63} 个明文对，选取 $2^{78.65}$ 个结构，共有 $2^{110.65}$ 个明文，形成 $2^{141.65}$ 个明文对。对选择的明文，经过 12 轮的加密后，保留满足密文差分形式为 $(M_0(00*0),****,0,00\gamma0)$ 的密文对，剩余明密文对个数为 $2^{141.5}\times 2^{-80}=2^{61.65}$。

(2)恢复密钥

1)猜测 32 比特密钥 RK_{24}。依次猜测 8 比特 $RK_{24,l}(l=0,1,2,3)$，过滤数据对，剩余数据对 $2^{61.65}\times 2^{-32}=2^{29.65}$ 个。

2)根据性质 1，猜测 6 比特的 $RK_{3,(2,3)}$。先猜测 1 比特的 $RK_{3,3}$，过滤数据对，再猜测 5 比特 $RK_{3,2}$，过滤数据对，剩余明密文对 $2^{29.65}\times 2^{-16}=2^{13.65}$ 个。

3)根据性质 2，猜测 26 比特 RK_{25}。不需要过滤数据对。

4)最后猜测 $(RK_{22}\oplus WK_3)_2$，对每个数据对密钥剩余的概率为 2^{-8}，经过 $2^{13.65}$ 个剩余明密文对，72 比特密钥 RK_{25}，$RK_{24}, RK_{3,(2,3)}, (RK_{22}\oplus WK_3)_2$，剩余的错误密钥个数为

$$2^{72}\times(1-2^{-8})^{2^{13.65}}<1$$

即可恢复正确密钥。攻击的数据复杂度为 $2^{110.65}$，时间复杂度主要由第 3)和第 4)步决定：

$$3)\ 2\times\frac{1}{8}\times\frac{1}{12}\times 2^{64}\times 2^{13.65}=2^{72.07}$$

$$4)\ 2\times\frac{1}{8}\times\frac{1}{12}\times 2^{72}\times[1+(1-2^{-8})+\cdots(1-2^{-8})^{2^{13.65}}]=2^{74.42}$$

该攻击的计算复杂度约为 $2^{74.7}$ 次加密运算，存储复杂度为得到符合要求的明密文对所需的存储空间，为 $4\times 2^{61.65}=2^{63.65}$。

4.3 13 轮的 CLEFIA-128 的不可能差分攻击

将 9 轮不可能差分路径 $(0,000\alpha,0,0)\xrightarrow{9}(0,\beta0000,0)$ 应用到 3—11 轮，在前后各加 2 轮，攻击 13 轮 CLEFIA-128，恢

复 $RK_{25}, RK_{24}, (RK_{22} \oplus WK_3)_0, RK_0, RK_1, (RK_3 \oplus WK_1)_3$ 共 122 比特的密钥。如图 3 所示。

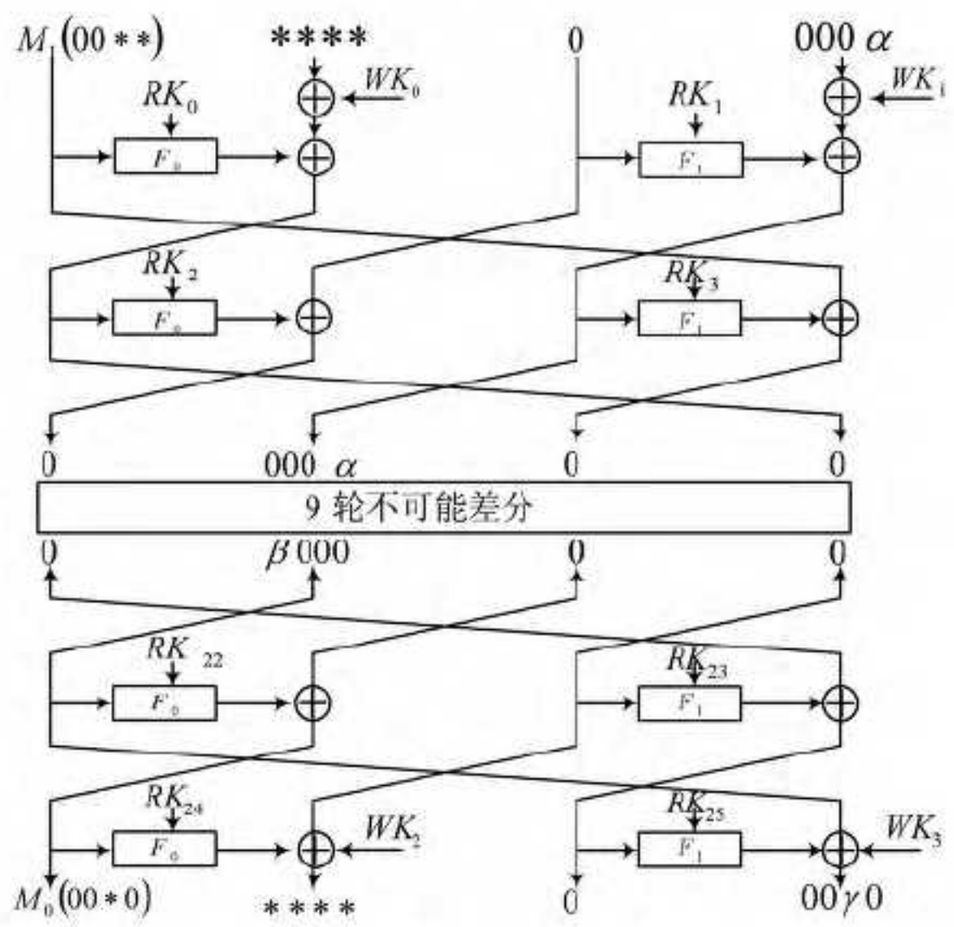

图 3　对 12 轮 CLEFIA-128 的不可能差分攻击

攻击过程：

(1)选择明文

令明文 $P=P_0|P_1|P_2|P_3$ 满足明文差分形式为 $(\Delta P_0, \Delta P_1, \Delta P_2, \Delta P_3)=(M_1(000*), ****, 0, 000*)$，该明文结构包括 2^{48} 个明文，可形成 2^{95} 个明文对。选取 2^N 个结构，共形成 2^{N+95} 个明文对。

(2)过滤数据对

这些明文对经过 13 轮加密，得到相应密文对，保留满足密文差分形式为 $(M_0(*000), ****, 0, 000*)$ 的密文对，则剩余明密文对个数为 $2^{N+95}\times 2^{-80}=2^{N+15}$。

(3)密钥恢复过程

1)恢复 RK_0。由于 P_0 和 P_0^* 已知，根据 S 盒差分分布表，第 1 轮 S 盒的输入差分和输出差分分别为 $M_1(000*)$，$M_0(****)$，由定理 1 经 1 次 F 运算，可计算 RK_0。

2)恢复 RK_{24}。由于密文 C_0^{13}, C_0^{13*} 已知，根据最后 1 轮 S 盒的输入差分和输出差分分别为 $M_0(*000), M_0(***)$，可经 1 次 F 运算计算 RK_{24}。

3)猜测 10 比特 RK_1，计算 $(RK_3 \oplus WK_1)_3$。根据性质 1，22 比特 RK_{24} 可由 RK_1 得出，只需猜测 10 比特 RK_{24} 即可。根据定理 2 可知由 RK_1 计算得到 $(RK_3 \oplus WK_1)_3$。

4)猜测 32 比特 RK_{25}，恢复 $(RK_{22} \oplus WK_3)_0$。根据第 2 轮的 S 盒输入差分为 $\beta 000$，输出差分为 $*000$，由定理 2 可计算 8 比特 $(RK_{22} \oplus WK_3)_0$。

如果存在一个明密文对使其能够保留下来，则删除猜测的 RK_1 和 RK_{25}。经过 2^{N+15} 个明密文对，若使剩余错误密钥个数为 $2^{122}\times(1-\frac{2^{42}}{2^{122}})^{2^{N+15}}=2^{80}$，则 $N=69.87$，数据复杂度为 $2^{117.87}$，时间复杂度为 $2^{69.87}\times 2^{15}\times 2^{42}=2^{126.87}$ 次 F 运算，大约为 $2^{122.17}$ 次加密运算。CLEFIA 加密运算用表格来标记被排除的子密钥，初始化时将表中的元素都置为 0，用 1 表示对应的子密钥被删除。对每个猜测的 RK_1 和 RK_{25}，RK_0，RK_{24}，$(RK_{22} \oplus WK_3)_0$，$(WK_1 \oplus RK_3)_3$ 这 80 比特被标记，占用内存 2^{80} 比特 $<2^{80}$ 组块，但仍需 2^{80} 组块存储剩余的候选密钥。

4.4　对 14 轮的 CLEFIA-256 改进攻击

根据孙兵在文献[6]中的 14 轮 CLEFIA-256 的攻击情况，给出改进结果。文献[6]中数据复杂度为 2^{112}，时间复杂度为 $2^{248.5}$，本文数据复杂度为 $2^{104.23}$，时间复杂度为 $2^{221.5}$ 次加密。

第 1 步　选取 $2^{40.23}$ 个结构，对输入差分满足 $(\Delta P_0, \Delta P_1, \Delta P_2, \Delta P_3)=(M_1(00**), ****, 0, 00**)$ 的明文对，加密 13 轮，保留满足输出差分为 $(\Delta C_0, \Delta C_1, \Delta C_2, \Delta C_3)=(M_1(00**), ****, 00*0, M_0(00*0))$ 的密文对，剩余 $2^{40.23}\times 2^{64}\times 2^{63}\times 2^{-40}=2^{127.23}$ 个明文对。

第 2 步　猜测 $RK_{27}|RK_1|RK_{25} \oplus WK_2$（共 12 个字节），计算 $RK_0|(RK_3 \oplus WK_1)_{2,3}|RK_{26}|RK_{24} \oplus WK_3|RK_{22,2}$（共 15 个字节），经过 $2^{127.23}$ 个明密文对后，剩余错误密钥数目为

$$(2^{216}-1)(1-\frac{2^{96}}{2^{216}})^{2^{127.23}}<1$$

所以攻击的数据复杂度为 $2^{104.23}$，时间复杂度主要由密钥恢复阶段决定。由于猜测 2^{96} 个密钥，需要计算 $2^{98}\times 2^{127.23}=2^{225.23}$ 次 F 运算，约合 $2^{221.5}$ 次加密运算。

表 1 显示了 11 轮 CLEFIA 算法的不可能差分攻击的结果，表 2 显示了 CLEFIA-128 的不可能差分攻击结果比较。改进的 14 轮的 CLEFIA-256 的结果比原来数据复杂度降低了 2^8，时间复杂度比原来降低了 2^{27} 次加密。

表 1　11 轮的 CLEFIA 算法不可能差分分析的结果比较

攻击轮数	恢复密钥长度	数据复杂度	时间复杂度	参考文献
11	64	$2^{118.5}$	$2^{66.5}$	[4]
11	72	2^{119}	2^{73}	[5]
11	72	$2^{110.7}$	2^{73}	本文
11	96	$2^{103.1}$	$2^{98.1}$	[4]

表 2　CLEFIA-128 算法不可能差分分析的结果比较

攻击轮数	数据复杂度	时间复杂度	存储复杂度	参考文献
12	$2^{118.9}$	2^{119}	2^{73}	[5]
12	$2^{110.93}$	2^{111}	—	[6]
12	$2^{110.65}$	$2^{74.7}$	$2^{63.65}$	本文
13	$2^{119.4}$	$2^{125.52}$	$2^{119.4}$	[7]
13	$2^{117.8}$	$2^{121.2}$	$2^{86.8}$	[8]
13	$2^{117.87}$	$2^{122.17}$	2^{80}	本文

结束语　本文利用已有的 9 轮不可能差分路径，分别使用 S 盒差分分布表和过滤数据对的方法，分析了 11 轮的 CLEFIA、14 轮的 CLEFIA-256，以及 12 轮和 13 轮的 CLEFIA-128，结果证明攻击都是有效的。接下来，需要考虑的问题是如何用不可能差分分析方法攻击相类似的算法，目前轻量级分组密码越来越受关注，它也将成为下一步工作的重点。本文参考文献[10]完成。

参 考 文 献

[1] Biham E, Biryukov A, Shamir A. Cryptanalysis of Skipjackreduced to 31 rounds using impossible differentials[C]// Jacques Stern. Advances in ryptology-Eurocrypt' 99. Czech Republic: Springer Berlin Heidelberg, 1999: 12-23

[2] Sony Corporation. The 128-bit Block cipher CLEFIA: Security and Performance Evaluations, Revision 1.0[R]. Tokyo: Sony Corporation, 2007

[3] Shirai T, Shibutani K, Akishita T, et al. The 128-bit block cipher CLEFIA[C]// Alex Biryukov. FSE 2007, LNCS 4593, Luxembourg: Springer Berlin Heidelberg, 2007: 181-195

[4] 王薇. 分组密码 CLEFIA 与基于四圈 AES 的消息认证码的安全性分析[D]. 济南:山东大学,2009

[5] Tsunoo Y, Tsujihara E, Shigeri M, et al. Impossible differential Cryptanalysis of CLEFA[C]// Kaisa Nyberg. FSE 2008, LNCS 5086. Lausanne, Switzerland: Springer Berlin Heidelberg, 2008: 398-411

[6] 孙兵. 分组密码的分析方法及应用研究[D]. 长沙:国防科学技术大学,2009

[7] Tang X, Sun B, Li R, et al. Impossible differential cryptanalysis of 13-round CLEFIA-128[J]. Journal of Systems and Software, 2011, 84(7): 1191-1196

[8] Mala H, Dakhilalian M, Shakiba M. Impossible differential attacks on 13-round CLEFIA-128[J]. Journal of Computer Science and Technology, 2011, 26(4): 744-750

[9] 吴文玲, 张文涛. 分组密码的设计与分析[M]. 北京:清华大学出版社, 2009: 68-72

[10] 刘青, 卫宏儒. 对完整轮数 ARIRANG 加密模式的新的相关密钥矩形攻击[J]. 计算机科学, 2013, 40(8): 109-114

(上接第 346 页)

的文件进行操作了,之后再编辑 /data/local.prop 文件,将里面的参数 ro.kernel.qemu 的值由 0 改为 1。ro.kernel.qemu 值代表 Android 厂商调试模式,在 Android 设备出厂前厂商需要对设备的各项功能进行调试,因而需要开放 root 权限,将这个值设为 1,出厂后这个值设为 0。ADB Restore 漏洞将重新把这个值设为 1,使手机就会处于调试模式,这样 adb shell 就取得了 root 权限。

在手机 app 中执行了相关的 shell 命令,比如 ln -s /data /data/local/tmp,得到的结果是:Permission denied。证明普通的用户 shell 无法完成这样的操作,所以这个漏洞也只有在 adb 条件下才能利用。

必须通过 adb 工具来利用的漏洞,黑客也可以利用其来进行攻击,利用的关键就是手机 app 必须取得 adb shell 的权限。这样的思路是具有可行性的,现在有很多的无线 adb 工具可以使手机不必通过 USB 线连接至电脑来接受电脑端程序的 adb 调试命令。黑客可以通过无线 adb 的场景,对手机进行调试,触发 root 漏洞,获取最高权限。接下来的工作中,我们会针对这样的攻击场景进行模拟实现,评估由此带来的危害,提出预防这样的攻击的办法。

结束语　在恶意软件利用 root 漏洞进行攻击时,因为 app 是最容易安装进手机的,一般的攻击手段为将 root 漏洞通过重打包的方式装进一个看似无害的 app 中,然后在用户安装后触发实施攻击,所以其在手机上直接触发的漏洞危害性较大,这一点在文献[4]的工作中得到了体现。

越来越多的研究者意识到了 root 漏洞对 Android 系统的安全产生的威胁,但是目前对 Android 系统的漏洞的研究还处于开始阶段。本文详细梳理了现有的 Android root 漏洞,对每个典型漏洞的特点进行了详细的描述,并且依据漏洞的利用方式进行了分类,为进一步开发这些漏洞的检查机制打下了基础。

在制定检测策略的时候,首先要注意的就是那些可以直接在手机上触发、适用的 Android 版本比较新的漏洞,比如 Linux 内核提权漏洞的 Android 平台移植版,以及一些基于 OEM 厂商硬件的漏洞。其次是那些适用 Android 版本比较低,但是利用起来比较方便的漏洞,如 Rageagainstthecage 和 GingerBreak 也要制定针对它们的检测机制,一方面是兼顾低版本 Android 用户,另一方面是防止一些第三方 ROM 在新版本 Android 系统上重新开放这些漏洞。

参 考 文 献

[1] Zhang Q, Li X, Yu X, et al. ASF: Improving Android Security with Layered Structure Instrumentation[M]// Contemporary Research on E-business Technology and Strategy. Springer Berlin Heidelberg, 2012: 147-157

[2] Ongtang M, McLaughlin S, Enck W, et al. Semantically rich application-centric security in Android[J]. Security and Communication Networks, 2012, 5(6): 658-673

[3] Bartel A, Klein J, Le Traon Y, et al. Automatically securing permission-based software by reducing the attack surface: An application to Android[C]// 2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2012: 274-277

[4] Zhou Y, Jiang X. Dissecting Android malware: Characterization and evolution[C]// 2012 IEEE Symposium on Security and Privacy (SP). IEEE, 2012: 95-109

[5] Jang W J, Cho S W, Lee H W, et al. Rooting attack detection method on the Android-based smart phone[C]// 2011 International Conference on Computer Science and Network Technology (ICCSNT). IEEE, 2011, 1: 477-481

[6] SEAndroid[OL]. http://selinuxproject.org/page/SEAndroid

[7] Android fragmentation[OL]. http://www.webopedia.com/TERM/F/fragmentation.html

[8] XDA[OL]. http://forum.xda-developers.com

[9] 福布斯中文网[EB/OL]. 201206 恶意软件可借由充电器入侵 iPhone 手机, http://www.forbeschina.com/review/201306/0026176.shtml

[10] KingRoot[OL]. http://www.pc6.com/az/75398.html

[11] SuperOneclick[OL]. http://luozhihao.wodemo.com/file/92858

[12] Z4Root http://forum.xda-developers.com/showthread.php?t=833953

[13] Framaroot[OL]. http://forum.xda-developers.com/showthread.php?t=2130276

[14] Rageagainstthecage [OL]. https://github.com/bibanon/Android-development-codex/wiki/rageagainstthecage

[15] GingerBreak [OL]. http://c-skills.blogspot.com/2011/04/yummy-ymuuy-gingerbreak.html

[16] CVE-2012-0056[OL]. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-0056

[17] CVE-2013-2094[OL]. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-2094

[18] Root exploit on Exynos [OL]. http://forum.xda-developers.com/showthread.php?t=2048511

[19] CVE-2013-2596[OL]. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-2596

[20] Revolutionary-zergRush local root 2.2/2.3. http://forum.xda-developers.com/showthread.php?t=1296916

[21] ADB Restore[OL]. http://forum.xda-developers.com/showthread.php?t=1439429