DES 差分攻击

1. 实验目的

很简单,我们的目的就是实现 DES 的三轮差分攻击。

2. 实验原理

2.1. 基本概念

首先为了明白 DES 差分攻击的原理，我们先来介绍几个简单的概念。

- 概念 1

设 $S_j\ (1\le j\le 8)$ 是一个特定的 S 盒，考虑长度 为 6 的有序对 (B_j, B_j^*),我们称 S_j 的输入异或为 $B_j \oplus B_j^*$,六位；输出异或为 $S(B_j)\oplus S(B_j^*)$，四位。

显然，如果我们构造一个输入异或到输出异或的函数，这并不是一个一一映射，会有不同的输入异或对应相同的输出异或，还有可能有的 4 位的比特串没有被映射到。

- 概念 2

对于任意一个 $B_j' \in (Z_2)^6 = \{(a_0,a_1,a_2,a_3,a_4,a_5) \mid a_j \in \{0,1\}\}$，我们可以定义集合 $\Delta(B_j)$ 为输入异或值为 B_j 的无序对。

注意 1：

如果不考虑顺序的话，显然对于任意一个异或来说，这个集合的大小为 64.

注意 2：

对于这个集合中的每一对元素，我们都可以利用表来列出每一对的输出异或。这样就会产生 64 到 16 的映射。这就是差分攻击的核心之处。

- 概念 3

对于长度为 6 的比特串 $B_j'(1\le j\le 8)$ 和长度为 4 的比特串 $C_j'(1\le j\le 8)$,我们可以做如下定义

$$IN_j(B_j', C_j') = \{B_j \in (Z_2)^6 \mid S_j(B_j)\oplus S_j(B_j \oplus B_j') = C_j'\}$$

这个集合定义的是什么呢？是输入异或为 B_j'，输出异或为 C_j' 的 6 比特串所在的集合，为什么不是对呢？是因为这是对称的。这个集合的元素必然为偶数。

这里，我们不妨定义 $N_j(B_j', C_j') = \left|IN_j(B_j', C_j')\right|$ 是上述集合的大小。

- 概念 4

在得到概念 4 之前，我们先来回顾一下之前所讲的一点知识。

我们可以知道第 i 轮中 S 盒的输入为 $B=E\oplus J$，其中 $E=E(R_{i-1})$ 是 R_{i-1} 的扩展。$J=K_i$ 是第 i 轮的密钥比特串。

那么我们可以对所有的 S 盒的输入异或计算如下

$$B\oplus B^*=(E\oplus J)\oplus(E^*\oplus J)=E\oplus E^*=E'$$

由此可见，输入异或并不依赖于密钥比特串 J，这一点也是非常重要的。也就是说，其实我们只需要扩展运算的结果就可以了。

那么假设我们知道 E_j 和 E_j^* 的值，以及 S_j 的输出异或值 $C_j'=S_j(B_j)\oplus S_j(B_j^*)$。又有 $E_j'=E_j\oplus E_j^*=B_j\oplus B_j^*$，也就是说扩展输入的异或值就是 S 盒的输入异或值，故必有

$$E_j\oplus J_j\in IN_j(E_j',C_j')$$

为了得到密钥 J，必须再做如下定义：

设 E_j 和 E_j^* 是长度为 6 的比特串，可以视为 DES 右半部分两个扩展后的一小部分，C_j' 为长度为 4 的比特串，定义

$$Test_j(E_j,E_j^*,C_j')=\{B_j\oplus E_j|B_j\in IN_j(E_j',C_j')\}$$

显然，$B_j\oplus E_j$ 是 J_j，E_j 固定，$E_j'=E_j\oplus E_j^*=B_j\oplus B_j^*$。

注意：

这里如果我们假设 E_j 和 E_j^* 是 DES 中第 j 个 S 盒的输入的前置，也就是还没有与密钥异或的那一部分的 6 比特串，并假设 S_j 的输出异或是 C_j'，记 $E_j'=E_j\oplus E_j^*=B_j\oplus B_j^*$.

如果我们已知扩展运算给定的输入 E_j，并且已知输入异或和输出异或，我们可能得到的输入必然属于 $IN_j(E_j',C_j')$，那么密钥 J_j 必然出现在 $Test_j(E_j,E_j^*,C_j')$ 中。

3. 实验流程

3.1.3 轮 DES 流程

首先，为了分析得到算法的流程，我们必须还得作如下分析：

这里，为了避免麻烦，我们的 DES 加密并没有最初的置换操作那一步，也没有最后的交换的那一步。

假设起始为 L，R

(一) 初始置换之后为 L_0,R_0

(二) 第一轮

密钥 K_1

$L_1=R_0$

$R_1=L_0 \oplus F(R_0,K1)$

(三) 第二轮

密钥 K_2

$L_2=R_1= L_0 \oplus F(R_0,K1)$

$R_2= L_1 \oplus F(R_1,K2)= R_0 \oplus F(R_1,K2)$

(四) 第三轮

密钥 K_3

$L_3=R_2= R_0 \oplus F(R_1,K2)$

$R_3=L_2 \oplus F(R_2,K3)= L_0 \oplus F(R_0,K1) \oplus F(R_2,K3)$

批注 [r1]: 我们攻击的是第 3 轮

3.2. 差分攻击分析

假定我们选择了两组明密文对：

明文对：**L_0R_0 ,$L^*_0R^*_0$ (经过置换之后的)**

密文对：**L_3R_3 ,$L^*_3R^*_3$**

显然有

$R_3= L_0 \oplus F(R_0,K1) \oplus F(R_2,K3)$

$R^*_3= L^*_0 \oplus F(R^*_0,K1) \oplus F(R^*_2,K3)$

如果我们选择 **$R_0=R^*_0$**,那么

$R'_3= R_3 \oplus \ R^*_3=L'_0 \oplus F(R_2,K3) \ \oplus F(R^*_2,K3)$

进而

$R'_3 \oplus L'_0=F(R_2,K3) \ \oplus F(R^*_2,K3)$

与此同时，$F(R_2,K3)$和 $F(R^*_2,K3)$分别是 P(C)和 $P(C^*)$是两个 S 盒经过置换后输出。而 P 是公开的。又我们可以简单的证明 $P(C) \ \oplus P(C^*)=P(C \oplus C^*)$,证明如下：

由于 P 是一个置换，那么此时我们对 P(C)的第 i 位与 $P(C^*)$的第 i 位，不放假设其原来是相应的第 j 位，那么 $P(C)_i \oplus P(C^*)_i = P^{-1}(\mathrm{i})_C \oplus P^{-1}(\mathrm{i})_{C^*} = C_j \oplus C^*_j$，相当于此时第 i 位的值是原来第 j 位的值的异或。同时，如果我们先异或，再置换，即 $\begin{matrix} P(C_j \oplus C^*_j) = \mathrm{P(i)} \\ C_j \oplus C^*_j = P^{-1}\left(R'_3 \oplus L'_0\right) \end{matrix}$，也可以得到相应的值。

因此我们可以得到

$$R'_3 \oplus L'_0=F(R_2,K3) \ \oplus F(R^*_2,K3)= P(C) \ \oplus P(C^*)= \ P(C_j \oplus C^*_j)$$

故而

$$C_j \oplus C^*_j = P^{-1}\left(R'_3 \oplus L'_0\right)$$

此时我们得到了输出异或。

此外 R2=L3 和 **$R2^*= L^*_3$** 是已知的（它们是密文的一部分），因此，可用公开已知的扩展

函数 **E** 计算 **E=(L3)**和 **E*=E(L3*)。**

对于第三轮来说，这就是 S 盒的一部分输入。

此时输入异或也得到了。

于是我们可以开始进行攻击了。

3.3.3 轮差分攻击攻击流程

首先初始化八个数组每个矩阵大小 64,CountJ。

主要分为两大部分。

(一) 主 while 循环---寻找 48 位密钥

（1） 读入数据两对明密文对。

（2） 计数 num

（3） 先分别而得到 R3 ′ 和 L0 ′ 然后先得到输出异或。$C'=P^{-1}(R3'\oplus L0')$

（4） 计算 E=E(L3)和 E*= E(L3*)

（5） For i=1 to 8 do

a) 得到第 i 部分输出异或 c

b) 进行集合测试

i. 得到第 i 部分输入异或 e

ii. for b= 000000 to 111111

- 计算 b 异或 e 得到 b_1----->也就是可能的一个输入。
- 计算 b，b_1 经过第 i 个 S 盒之后的异或是不是 c
- 是的话，计数器相应位置,也就是可能的密钥处加 1

（6） 对每一个计数器阵列进行判断是否都只有一个数为 num，如果是的话就说明已经成功了。置标记退出。否则，继续循环。

(二) 得到 56 位密钥

（1） 首先先得到可以得到的密钥位

（2） 然后枚举剩余密钥位，右移得到最初的密钥，最后进行检测。

4. 实验工具及方式

编译器：eclipse。

程序：python。

5. 程序（参见源文件）

实验输出

数据 1：

明文 1：748502cd38451097
密文 1：03c70306d8a09f10
明文 2：3874756438451097
密文 2：78560a0960e6d4cb

明文 1：486911026acdff31
密文 1：45fa285be5adc730
明文 2：375bd31f6acdff31
密文 2：134f7915ac253457

明文 1：357418da013fec86
密文 1：d8a31b2f28bbc5cf
明文 2：12549847013fec86
密文 2：0f317ac2b23cb944