密码学实验报告 实验四

2019 年 3 月 27 日

1 DES 加解密算法

1.1 算法原理

DES 有两个输入：64 位明文和 56 位密文。首先 64 位的明文经过初始置换 IP 而被重新排列。然后进行 16 轮相同函数的作用，每轮作用都有置换和代替。最后一轮迭代的输出有 64 位，它是输入明文和密钥的函数。其左半部分和右半部分互换产生预输出。最后预输出再被与初始置换 IP 互逆的置换 IP^{-1} 作用产生 64 位的密文。除了初始和末尾的置换，DES 的结构与 Feistel 密码结构完全相同。子密钥生成的步骤为：初始密钥经过一个置换，再经过循环左移和一个置换分别得到各轮的子密钥K_i用于各轮的迭代。每轮的置换函数都一样，但是由于密钥的循环移位使得各轮子密钥互不相同。

1.2 算法实现的伪代码

DES 加密算法 利用 DES 加密

输入：64 位明文P，64 位密钥K
输出：64 位密文C

function $des\_encrypt(K, P)$
    对K做 PC1 置换
    令C_0, D_0分别为密钥K的高 28 位和低 28 位
    for $i \leftarrow 1$ to 16 **do**
        将C_{i-1}, D_{i-1}分别循环左移一位或两位得到C_i, D_i
        令K_i为$C_i || D_i$
        对K_i做 PC2 置换
    end for
    对P做初始 IP 置换
    令L_0, R_0为明文的高 32 位和低 32 位
    for $i \leftarrow 1$ to 16 **do**
        令$extR_i$为R_{i-1}做扩展置换的结果
        计算$extR_i$的K_i异或
        令SIN_j为第j个 S 盒的输入
        for $j \leftarrow 1$ to 8 **do**

将SIN_j的第一位和最后一位作为行，中间四位作为列

选择第j个 S 盒中对应位置的数作为输出$SOUT_j$

end for
对$SOUT$做 P 置换
令R_i为L_{i-1}与上述结果的异或值
令L_i等于R_{i-1}
end for
令C为$R_{16}||L_{16}$
对C做IP^{-1}置换
return C
end function

DES 解密算法 利用 DES 解密
输入：64 位密文C，64 位密钥K
输出：64 位明文P
function $des\_decrypt(K, C)$
解密只需把子密钥使用次序反转，其余与加密相同
return P
end function

1.3 测试样例及运行结果

我们假设密钥$K = (133457799BBCDFF1)_{16}$，尝试对明文

$$P = (1234567890FEABCD)_{16}$$

进行加密，得到密文

$$C = (9789C520777B83CE)_{16}$$

再使用相同的密钥对密文解密，得到相同的明文，证明算法正确。

2 DES 差分攻击算法

2.1 算法原理

首先我们对于 DES 的分析可以忽略初始置换 IP 和 IP^{-1}。并且可以对 DES 限制到n轮$(n \leq 16)$。以L_0, R_0为明文，L_n, R_n为对应的密文。对两个明文L_0R_0和$L_0^*R_0^*$，规定它们的异或值$L_0'R_0' = L_0R_0 \oplus L_0^*R_0^*$。

设S_j为第j个 S 盒$(1 \leq j \leq 8)$，考虑 6bits 的有序对(B_j, B_j^*)，规定S_j的输入异

或为$B_j \oplus B_j^*$，输出异或为$S(B_j) \oplus S_j(B_j^*)$。同时，对$\forall B_j' \in (Z_2)^6$，定义集合$\Delta(B_j')$是由输入异或为B_j'的有序对(B_j, B_j^*)组成。

可以通过$\Delta(B_j') = \{(B_j, B_j \oplus B_j') | B_j \in (Z_2)^6\}$来计算$2^6 = 64$个有序对元素，并且对每一对都可以计算出$S_j$的输出异或，同时用表列出结果的分布。有 64 个输入异或，经S_j计算的输出分布在$2^4 = 16$中可能的结果之中，且这个分布是非均匀的。称一个 S 盒的所有可能有序对的输入异或和输出异或分布表为该 S 盒的差分分布表。

通过差分分布表，我们可以定义

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

$$N_j(B_j', C_j') = |IN_j(B_j', C_j')|$$

对于 DES 的 8 个 S 盒的每个盒有 64 种可能的输入异或，于是可以算出有 512 个分布。在第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。

可以对 8 个 S 盒的输入异或计算如下：

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

若我们已知E_j和E_j^*的值和S_j的输出异或$C_j' = S_j(B_j) \oplus S_j(B_j^*)$，必有

$$E_j \oplus J_j \in IN_j(E_j', C_j') \quad B_j = E_j \oplus J_j \quad B_j' = E_j \oplus E_j^*$$

我们的目的是要破译密钥J的部分比特串J_j，定义测试集合

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

可以得到以下定理：

假设E_j和E_j^*是 S 和S_j的两个输入，S_j输出异或为C_j'，记$E_j' = E_j \oplus E_j^*$，那么密钥比特J_j出现在集合$Test_j(E_j, E_j^*, C_j')$之中，共有$N_j(E_j', C_j')$个可能。

如果我们有两个这样的三元组E_j, E_j^*, C_j'，得到集合$Test_j^{(1)}, Test_j^{(2)}$，则密钥比特$J_j \in Test_j^{(1)} \cap Test_j^{(2)}$，若有足够多的三元组，则一定能确定出$J_j$的具体值。

在 3 轮 DES 中，使用明文对和相应的密文对进行分析

明文对为：$L_0R_0, L_0^*R_0^*$

密文对为：$L_3R_3, L_3^*R_3^*$

有表达式

$$R_3 = L_2 \oplus F(R_2, K_3) = R_1 \oplus F(R_2, K_3) = L_0 \oplus F(R_0, K_1) \oplus F(R_2, K_3)$$

类似地

$$R_3^* = L_0^* \oplus F(R_0^*, K_1) \oplus F(R_2^*, K_3)$$

可以得到

$$R_3' = L_0' \oplus F(R_0, K_1) \oplus F(R_0^*, K_1) \oplus F(R_2, K_3) \oplus F(R_2^*, K_3)$$

假设选择了明文使$R_0 = R_0^*$，则$R_0' = 0$且$F(R_0, K_1) = F(R_0^*, K_1)$，所以

$$R_3' = L_0' \oplus F(R_2, K_3) \oplus F(R_2^*, K_3)$$

由于R_3', L_0'已知，可以得到

$$F(R_2, K_3) \oplus F(R_2^*, K_3) = R_3' \oplus L_0'$$

设C, C^*为 S 盒的两个输出，则 F 函数的输出

$$F(R_2, K_3) = P(C) \quad F(R_2^*, K_3) = P(C^*)$$

由于 P 置换为已知的，因此

$$P(C) \oplus P(C^*) = R_3' \oplus L_0'$$

由于置换的性质，我们有

$$C' = C \oplus C^* = P^{-1}(R_3' \oplus L_0')$$

为第三轮中 S 盒的输出异或

另外$R_2 = L_3, R_2^* = L_3^*$已知，可以用公开的扩展置换 E 计算

$$E = E(L_3) \quad E^* = E(L_3^*)$$

对第三轮来说，它们是 S 盒的输入，可由E, E^*, C'构造 Test 集合，来确定第三轮密钥K_3中的 48bits。56bits 原密钥剩下的 8bits 可通过穷举$2^8 = 256$中可能确定。

2.2 算法实现的伪代码

三轮 DES 差分攻击算法 利用差分方法对三轮 DES 进行攻击

输入：选择明文对组P，对应密文对组C

输出：密钥K

function $diffcrypt(P, C)$

for $j \leftarrow 1\ to\ 8$ **do**
for $bj \leftarrow 1\ to\ 2^6$ **do**
for $bi \leftarrow 1\ to\ 2^6$ **do**
$ci \leftarrow sbox_j(bi) \oplus sbox_j(bi \oplus bj)$
$Test_j(bj, ci) = bi$
end for
end for
end for
建立大小为 48 的计数器阵列
for $i \leftarrow 1\ to\ length(P)$ **do**
令P_i为$L_0R_0, L_0^*R_0^*$，C_i为$R_3L_3, R_3^*L_3^*$，其中$R_0 = R_0^*$
计算$R_3' = R_3 \oplus R_3^*, L_0' = L_0 \oplus L_0^*$
计算$C' = P^{-1}(R_3' \oplus L_0')$
计算$E = E(L_3), E^* = E(L_3^*)$
for $j \leftarrow 1\ to\ 8$ **do**
计算$Test_j(E_j, E_j^*, C_j')$
end for
对应计数器数字+1
end for
令密钥K_3为计数器阵列中计数为$length(P)$的值
将密钥循环右移至初始密钥K_0
遍历剩余的2^8种可能恢复为$56bits$密钥K
return K
end function

2.3 测试样例及运行结果

我们假设密钥为$K = (FE326232EA6D0D73)_{16}$，使用两对选择明文

$$P = \begin{Bmatrix} (748502CD38451097)_{16}, (3874756438451097)_{16} \\ (357418DA013FEC86)_{16}, (33549847013FEC86)_{16} \end{Bmatrix}$$

用 3 轮 DES 进行加密，得到密文

$$C = \begin{Bmatrix} (463145E6EB5126B1)_{16}, (A3A9B5104AE17F2A)_{16} \\ (94E8B4BEA3346EF2)_{16}, (CD661D83DE4AF7E8)_{16} \end{Bmatrix}$$

使用上述明密文对 3 轮 DES 进行差分攻击，得到密钥

$$K = (FE326232EA6D0D73)_{16}$$

证明算法正确。

3 三重 DES 加解密算法

3.1 算法原理

三重 DES 使用了两个不同的密钥进行三次加密，可以提升已知明文的攻击代价，防止中间相遇攻击，具体运算过程是加密-解密-加密（EDE），式子为

$$C = E\left(K_1, D\left(K_2, E(K_1, P)\right)\right)$$

$$P = D\left(K_1, E\left(K_2, D(K_1, C)\right)\right)$$

对 3DES 的穷举攻击的代价是$2^{112} \approx (5 \times 10^{33})$数量级，用差分密码分析的代价是按指数增长的，与单 DES 比较超过10^{52}。

3.2 算法实现的伪代码

3DES 加密算法 利用三重 DES 加密

输入： 64 位明文P，64 位密钥K_1, K_2
输出： 64 位密文C

```
function 3des_encrypt(P, K1, K2)
    C ← des_encrypt(K1, P)
    C ← des_decrypt(K2, C)
    C ← des_encrypt(K1, C)
    return C
end function
```

3DES 解密算法 利用三重 DES 解密

输入： 64 位密文C，64 位密钥K_1, K_2
输出： 64 位明文P

```
function 3des_encrypt(C, K1, K2)
    P ← des_decrypt(K1, C)
    P ← des_encrypt(K2, P)
    P ← des_decrypt(K1, P)
    return P
end function
```

3.3 测试样例及运行结果

我们假设密钥

$$K_1 = (133457799BBCDFF1)_{16}$$
$$K_2 = (0E329232EA6D0D73)_{16}$$

尝试对明文

$$P = (1234567890ABCDEF)_{16}$$

进行加密，得到密文

$$C = (684388A3575C0137)_{16}$$

再使用相同的密钥对密文解密，得到相同的明文，证明算法正确。

4 两重 S-DES 的中间相遇攻击算法

4.1 算法原理

对于双重 DES 加密，使用两个密钥K_1, K_2对给定明文P进行加密

$$C = E\big(K_2, E(K_1, P)\big)$$

$$P = D\big(K_1, D(K_2, C)\big)$$

对于 DES，这种方法的密钥长度为$56 \times 2 = 112$位，密码强度增加了，且该映射不能被单 DES 所定义。

但是我们可以使用中间相遇攻击方法对其进行攻击，假设

$$C = E\big(K_2, E(K_1, P)\big)$$

则有

$$X = E(K_1, P) = D(K_2, C)$$

给定明密文对(P, C)，首先将P按所有可能的密钥K_1加密，得到的2^{56}个结果存入一个表内，然后将C用所有可能的密钥K_2解密，每解密一次就将解密结果与表中的值比较，如果有相等的，就用刚才测试的两个密钥对一个新的明密文对进行验证，如果两个密钥产生了正确的密文，就认定这两个密钥为正确的密钥。

对任意给定的明文P，采用双重 DES 加密后可能得到2^{64}个密文中的一个，双重 DES 使用了112位密钥，所以密钥空间为2^{112}。因此对明文P，可产生密文C的密钥个数平均为$2^{112}/2^{64} = 2^{48}$。故上述攻击过程对第一个(P, C)对将产生2^{48}个错误的结果，而对第二个 64 位的明密文对(P, C)，错误结果的概率就将为$2^{48-64} = 2^{-16}$。也就是说中间相遇攻击使用两组已知明密文对就可以得出正确结果的概率为$1 - 2^{-16}$。结论是：已知明文攻击可以成功对付密钥长度为112位的双重 DES，其付出是2^{56}数量级的，比攻击单 DES 所需的2^{55}数量级多不了多少。

4.2 算法实现的伪代码

中间相遇攻击算法 对两重 S-DES 进行中间相遇攻击

输入： 已知明密文对(P, C)

输出： 密钥K_1, K_2

function $meet\_in\_the\_middle(P, C)$
  for $K_1 \leftarrow 1 \ to \ 2^{10}$ **do**
    $table[sdes\_encrypt(K_1, P)] \leftarrow K_1$
  end for
  for $K_2 \leftarrow 1 \ to \ 2^{10}$ **do**
    $X \leftarrow sdes\_decrypt(K_2, C)$
    if X in $table$ **then**
      if $C' = 2sdes\_encrypt(table[X], K_2, P')$ **then**
        return $table[X], K_2$
      end if
    end if
  end for
  return "攻击失败"
end function

4.3 测试样例及运行结果

我们假设密钥

$$K_1 = (1010000010)_2$$
$$K_2 = (1100101011)_2$$

用两重 S-DES 对四个已知明文进行加密

$$P = \begin{Bmatrix} (01000101)_2 \\ (11010101)_2 \\ (11100111)_2 \\ (10110111)_2 \end{Bmatrix}$$

得到密文

$$P = \begin{Bmatrix} (10010110)_2 \\ (11010100)_2 \\ (11110100)_2 \\ (10100100)_2 \end{Bmatrix}$$

使用上述明密文对两重 S-DES 进行中间相遇攻击，得到密钥

$$K_1 = (1010000010)_2$$
$$K_2 = (1100101011)_2$$

证明算法正确。

5 总结

我认为本次实验中的难点在于前两个实验：DES 加密算法和差分攻击方法的实现，为了完成这两个实验，我们需要对 DES 的原理以及差分攻击的原理掌握得十分透彻，并且实现这两种算法。

在 DES 算法的实现中，我首先查到 DES 的标准置换表和 S 盒，利用位运算的特性，实现了对比特串的基本操作方法，定义了 DES_base 类，继而通过继承基类实现 DES 类，编写了 16 轮 DES 的加解密算法，使用标准的样例对算法进行了正确性验证，证明我们的算法可以在很快时间内计算出结果，我们还实现了对文件的加解密操作，并且也得到了实际验证。

在 DES 的差分攻击算法中，我重新认真学习了差分攻击的原理，编写出了对 3 轮 DES 的攻击算法，在验证中使用两组选择明密文对即可在 1s 内破解出正确的密钥，若使用更多的明密文对，则可以更快地破解。

在实现三重 DES 时，我们直接对原来的 DES 算法进行了扩展，很容易地实现了三种 DES 的加解密。

最后的两重 S-DES 中间相遇攻击算法中，我们首先实现了 S-DES 加解密方法，然后按照中间相遇攻击的原理对两重 S-DES 进行攻击，在实际验证中我发现了一些问题，由于 S-DES 的密钥长度为 10bit，明文长度为 8bit，若仅使用两组已知明密文，平均会得到2^4种可能的结果，通过这个来得到真正的密钥是不可行的，所以我们还需要增加已知明密文的数量，一般增加到 4 组时，即可通过中间相遇攻击得到正确的密文。

本次的实验让我们对课本上的知识有了很深刻的了解，在查阅资料时也学习到了不少课外知识，相信在之后编写 AES 算法中可以更容易地上手。