密码学实验报告 4

张天辰 17377321

2019 年 4 月 3 日

1 DES 算法

1.1 简介

DES 是针对 64bit 明文和 64bit 密钥的基于 Feistel 结构的分组加密方案。该算法先将密钥进行 PC1 变换后分成两半每次循环左移特定位数再进行 PC2 变换生成 16 个子密钥。然后将明文进行 IP 置换，每轮加密结果的左半部分就是上一轮的右半部分，右半部分是左部分与右部分经过 f 函数后异或的结果。最后再将得到的左右部分互换，并进行 IP 逆置换就得到密文。DES 加密算法与解密算法相同，只是子密钥使用顺序完全相反。

1.2 算法实现

Algorithm 1 DES

输入: 明文 $src\_message$（加密），密钥 src_key，密文 src_cipher（解密）

输出: 密文 $cipher$（加密），明文 $message$（解密）

function GENERATE_KEY(src_key)
    $bin\_key \leftarrow 64bit\ src\_key$
    $key \leftarrow \ PC1(bin\_key)$
    $keys \leftarrow key$ 左右两半进行特定次循环左移生成的 56bit 值
    $keys$ 每项进行 PC2 变换
    return $keys$
end function
function F($right, key$)
    $b \leftarrow key \oplus \ E(right)$
    $c \leftarrow b$ 分 8 部分进入 S 盒
    return 变换 $P(c)$
end function
function DES_ENCRYPT($src\_message, src\_key$)
    $keys \leftarrow$GENERATE_KEY(src_key)
    $message \leftarrow src\_message$ 转换为 64bit 二进制后再进行 IP 变换
    $left \leftarrow message[0:32]$

17: $right \leftarrow message[32:]$
18: **for** each $i \in [0, 16)$ **do**
19: $left, right \leftarrow right, left \oplus \text{F}(\text{right, keys[i]})$
20: **end for**
21: **return** 变换 $IP^{-1}(right + left)$
22: **end function**
23: **function** DES_DECRYPT(src_cipher, src_key)
24: $keys \leftarrow$GENERATE_KEY(src_key) 逆序
25: $cipher \leftarrow src\_cipher$ 转换为 64bit 二进制后再进行 IP 变换
26: $left \leftarrow cipher[0:32]$
27: $right \leftarrow cipher[32:]$
28: **for** each $i \in [0, 16)$ **do**
29: $left, right \leftarrow right, left \oplus \text{F}(\text{right, keys[i]})$
30: **end for**
31: **return** 变换 $IP^{-1}(right + left)$
32: **end function**

1.3 测试样例

```
                              $ python3 DES.py -m 0123456789abcdef -k 51636275abcddcba
DES ENCRYPTOR
====================
Your message is: 0123456789abcdef
Your key is: 51636275abcddcba
Encrypt finished. Your cipher is: 84241821bec4d186

Task completed.
                              $ python3 DES.py -c 84241821bec4d186 -k 51636275abcddcba
DES_DECRYPTOR
====================
Your cipher is: 84241821bec4d186
Your key is: 51636275abcddcba
Decrypt finished. Your message is: 0123456789abcdef
```

图 1: DES

2 DES 差分攻击

2.1 简介

DES 差分攻击的基础是，对于同一个 S 盒，不同输入异或的输出异或分布并不均匀，因此可以大大减少穷举范围。对于三轮 DES，最后一轮的输入异或可以从密文的左半部分经过 E 变换得到，这基于如下事实：

$$B_j \oplus B_j^* = (E_j \oplus K_3) \oplus (E_j^* \oplus K_3) = E_j \oplus E_j^*$$

输出异或可以由如下方式得到：

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

令 $L_0 = L_0^*$，则：

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

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

因此可以获得输入异或与输出异或。此后便可根据分布表找到对于特定输入异或和 S 盒的可能输入 B。根据 $E \oplus B = K$ 可以确定可能的 K_3。使用多个明密文对可以对可能的 K_3 集合取交集，可以确定 K_3。

得到 K_3 就可以确定原密钥的 48 位。剩余的 8 位可以由穷举攻击暴力破解得到。

2.2 算法实现

Algorithm 2 3 轮 DES 差分攻击

输入: 三组明密文组（每组两对）$couple_1, couple_2, couple_3$

输出: 唯一可能密钥 $final\_key$

1: **function** ATTACK($couple_1, couple_2, couple_3$)
2: 对 $couple_1, couple_2, couple_3$ 分别生成给定输入异或和 S 盒的输出异或关于输入的分布表
3: 对三组分别计算输入异或，然后查表得到可能的输入从而计算可能的密钥
4: 对密钥可能集合求交集，确定唯一可能密钥 $key48$
5: 寻找密钥所缺少的 8 位，并记录下标
6: 从 0 到 256 遍历缺少的 8 位，并加密验证，得到最终密钥 $final\_key$
7: **end function**

2.3 测试样例

上述算法及此样例获得的密钥都是原始 64 位密钥通过 PC1 变换后得到的 56 位密钥。它可以直接被使用到加密中，故不再做还原。

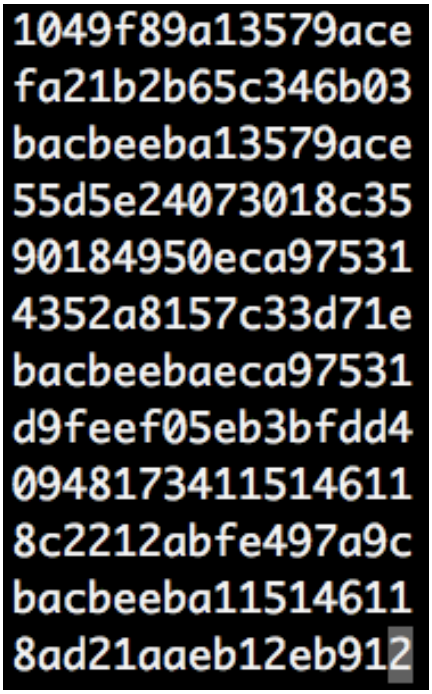

图 2: 测试用明密文对

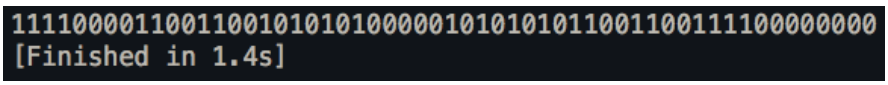

图 3: 输出的 56 位密钥

3 三重 DES 加解密

3.1 简介

三重 DES 提高了 DES 安全性，增加穷举攻击的范围。它使用两个密钥 key_1, key_2。加密过程为，先用 key_1 加密，再用 key_2 解密，再用 key_1 加密。三重 DES 完全兼容 DES 算法（只要让两个密钥相等）。

3.2 算法实现

Algorithm 3 三重 DES 加解密

输入: 明文 $src\_message$（加密），密钥 src_key1 src_key2，密文 src_cipher（解密）

输出: 密文 $cipher$（加密），明文 $message$（解密）

function 3DES_ENCRYPT($src\_message, src\_key1, src\_key2$)
    $cipher1 \leftarrow$DES_ENCRYPT($src\_message, src\_key1$)
    $cipher2 \leftarrow$DES_DECRYPT($cipher1, src\_key2$)
    $cipher3 \leftarrow$DES_ENCRYPT($cipher2, src\_key1$)
    return $cipher3$
end function
function 3DES_DECRYPT($src\_cipher, src\_key1, src\_key2$)
    $message1 \leftarrow$DES_DECRYPT(src_cipher, src_key1)
    $message2 \leftarrow$DES_ENCRYPT($message1, src\_key2$)
    $message3 \leftarrow$DES_DECRYPT($message2, src\_key1$)
    return $message3$
end function

3.3 测试样例

```
[redacted]$ python3 3DES.py -m 6243265362345676 --key1=5432153464572462 --key2=6226627427477721 -c 7431424327354314
3DES ENCRYPTOR
====================
Your message is: 6243265362345676
Your keys are: 5432153464572462    6226627427477721
Encrypt finished. Your cipher is: f2fd78c9809226f2

3DES_DECRYPTOR
====================
Your cipher is: 7431424327354314
Your keys are: 5432153464572462    6226627427477721
Decrypt finished. Your message is: 3c6bc2085b2d9166

Task completed.
```

图 4: 三重 DES

4 二重 S-DES 的中间相遇攻击

4.1 简介

二重 DES 存在中间相遇攻击的攻击方式。为了简化，采用 S-DES 作为示例。已知多个明密文对下，攻击流程如下：

首先，取一个明密文对，将其明文加密一次，密钥遍历所有 2^10 种可能，将所有加密结果存储起来。其次，取该明密文对的密文，将其解密一次，密钥遍历所有 2^10 种可能。如果加密结果与解密结果有重合的地方（谓之“中间相遇”），则得到相遇点的两个密钥就是可能的密钥。用其他已知的明密文对进行验证，就可以得到是否为真正密钥，从而实现攻击。

S-DES 密钥很弱，事实上即使是用所有可能的明文作为检验，依然可能得到不止一组密钥，然而作为攻击者，得到两三组密钥，任务可以算是完成了。

4.2 算法实现

因为 S-DES 是 DES 的简化版本，简化的地方仅仅在于明文和密钥长度减少，轮数减少，因此不再在这里重写 S-DES 加解密伪代码。

Algorithm 4 S-DES 的中间相遇攻击

输入: 三组明密文组（每组两对）$couple_1, couple_2, couple_3$

输出: 可能密钥对 $final\_key$

```
function S_DES_MITM(couple_1, couple_2, couple_3)
    for each key_1 ∈ [0, 1024) do
        input_dict[S-DES_ENCRYPT(couple1.message, key_1)].append(key_1)
    end for
    for each key_2 ∈ [0, 1024) do
        if S-DES_DECRYPT(couple_1.cipher) ∈ input_dict.keys() then
            maybe_key1 = input_dict[S-DES_DECRYPT(couple_1.cipher)]
            maybe_key2 = key_2
            for each k ∈ maybe_key1 do
                用另外的明密文对检验这两个密钥
                if 检验正确 then
                    final_key.append((k, maybe_key2))
                end if
            end for
        end if
    end for
    return final_key
end function
```

4.3 测试样例

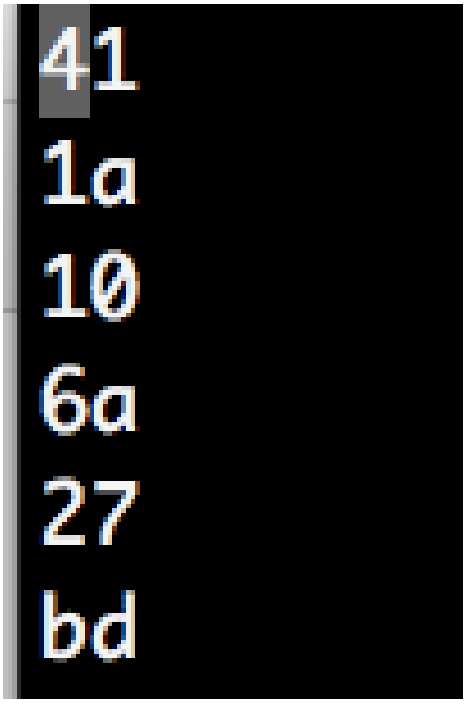

图 5: 测试用明密文对

```
The probable key pairs(in decimal):
(630, 945)
(598, 953)
(598, 1009)
(630, 1017)
[Finished in 0.9s]
```

图 6: 可能的密钥对

5 感想

诚然，仅凭上课就完全理解 DES 以及各种攻击对于我而言是不现实的，事实证明通过写代码，我轻松地记住了很多关于 DES 的细节和攻击原理。足见代码是用来加深理解的而不是为了“实现”。因此那些不能加深理解的代码就没什么用了。

因为这次作业设计对明密文对的一类操作，所以我尝试使用了面向对象的编程方式，又尝试了命令行参数或者文件读入的 IO 方法。事实证明这些方式能节省很多时间。这门课让我加强了编程能力，算是一个副产品吧。

总而言之，本次实验相比前几次实验，更让我觉得有趣。