实 验 报 告

课程名称	密码学基础
学　　院	计算机科学技术学院
专　　业	信息安全
姓　　名	冉津豪
学　　号	17307130179

开 课 时 间 **2019** 至 **2020** 学年第 二 学期

实验项目名　　称	DES&AES	成绩	

一、实验目的

1. Implementing DES algorithm
2. Understanding usage of AES algorithm

二、实验内容

1. DES

DES 是一种分组加密算法，该算法每次处理固定长度的数据段，称之为分组。DES 分组的大小是 64 位，如果加密的数据长度不是 64 位的倍数，可以用‘0’作填充位。DES 的安全性依赖于“混乱和扩散”的原则。混乱的目的是为隐藏任何明文同密文、或者密钥之间的关系，而扩散的目的是使明文中的有效位和密钥一起组成尽可能多的密文。两者结合到一起就使得安全性变得相对较高。具体步骤如下：

Step 1: Create 16 subkeys, each of which is 48-bits long.

将 64 位密钥经过 PC-1 置换后，输出 56 位密钥，并按高低 28 位分为左半部分 C0 和右半部分 D0 各 28 位。以 C0，D0 为初始状态生成子密钥，每一轮经过左移、合并、置换，共 16 轮，即生成 16 个子密钥。

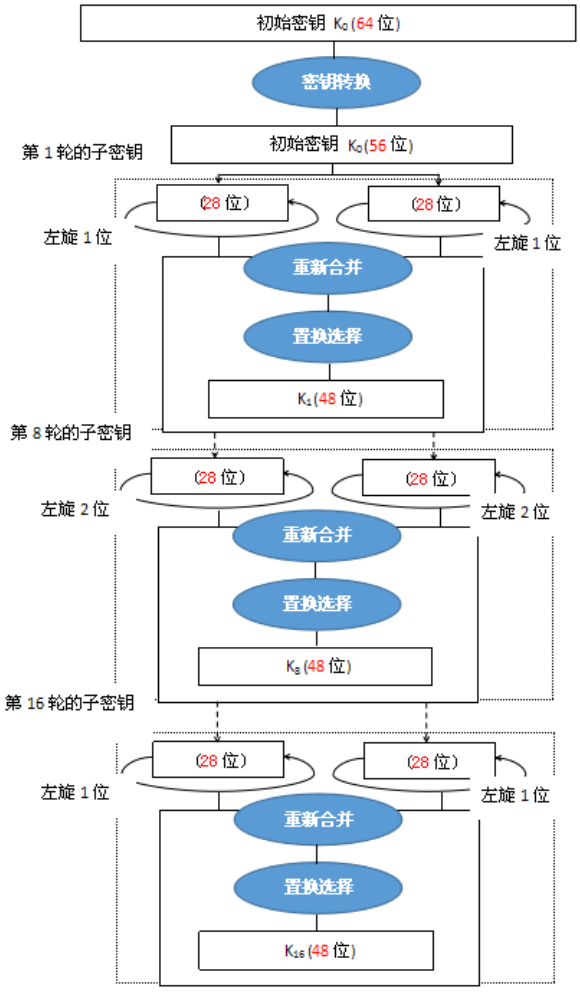

Figure 1

Step 2: Encode each 64-bit block of data.

对需要加密的数据进行分组后，对每一组数据的处理是相同的。首先对分组按照 IP 进行初始置换，并将输出按高低 32 位分为左半部分 L0 和右半部分 R0。此后将对 L0，R0 进行 16 轮的处理，下一轮的左半部分 L_{i+1} 就是 Ri，而 R_{i+1} 则是经过 E 置换，相应轮次密钥异或，S 盒替换，P 盒置换，L_i 异或后得到的。16 轮后将 L_{16}，R_{16} 交换、连接并做最后置换，得到最终 64 位的结果。

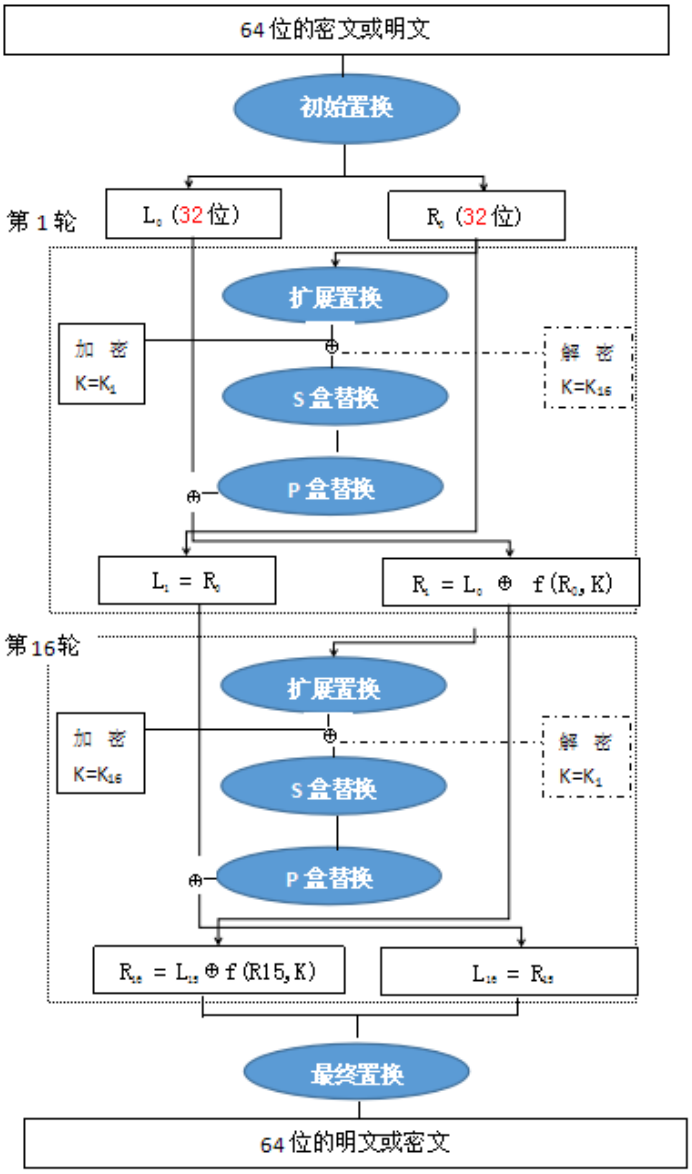

Figure 2

2. AES

高级加密标准(AES, Advanced Encryption Standard)为目前最常见的对称加密算法，被广泛应用于应用加密，网络通信加密。AES 同样为基于对称密钥的分组加密算法。

AES 加密过程是在一个 4×4 的字节矩阵上运作，这个矩阵又称为“体（state）”，其初值就是一个明文区块（矩阵中一个元素大小就是明文区块中的一个 Byte）。（Rijndael 加密法因支持更大的区块，其矩阵的“列数（Row number）”可视情况增加）加密时，各轮 AES 加密循环（除最后一轮外）均包含 4 个步骤：

1. AddRoundKey—矩阵中的每一个字节都与该次回合密钥（round key）做 XOR 运算；每个子密钥由密钥生成方案产生。
2. SubBytes—透过一个非线性的替换函数，用查找表的方式把每个字节替换成对应的字节。
3. ShiftRows—将矩阵中的每个横列进行循环式移位。
4. MixColumns—为了充分混合矩阵中各个直行的操作。这个步骤使用线性转换来混合每内联的四个字节。最后一个加密循环中省略 MixColumns 步骤，而以另一个 AddRoundKey 取代。

三、实验步骤

1. Implementing permutation_by_table function

在给定的参数调用中，block 及 key 都是整数的形式，所以所有的步骤都以整数形式处理。

```
1       def permutation_by_table(block, block_len, table):
2           '''block of length block_len permutated by table table'''
3           # WRITE YOUR CODE HERE!
4           res = 0
5           for i in range(0, len(table)):
6               res = (res << 1)+((block >> (block_len-table[i])) & 1)
7           return res
```

2. Create 16 subkeys, each of which is 48-bits long.

给定初始的 C0，D0，经过左移、合并、置换，共 16 轮，生成 16 个 48 位长的子密钥，按顺序存储在 round_keys 中。

```
1       def generate_round_keys(C0, D0):
2           '''returns dict of 16 keys (one for each round)'''
3           round_keys = dict.fromkeys(range(0,17))
4           lrot_values = (1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1)
5
6           # left-rotation function
7           lrot = lambda val, r_bits, max_bits: \
8           (val << r_bits%max_bits) & (2**max_bits-1) | \
9           ((val & (2**max_bits-1)) >> (max_bits-(r_bits%max_bits)))
10
11          # initial rotation
12          C0 = lrot(C0, 0, 28)
13          D0 = lrot(D0, 0, 28)
14          round_keys[0] = (C0, D0)
15
16          # create 16 more different key pairs
17          # WRITE YOUR CODE HERE!
18          for i in range(1,17):
19              (C, D) = round_keys[i-1]
20              Ci = lrot(C, lrot_values[i-1], 28)
```

```
            Di = lrot(D, lrot_values[i-1], 28)
            round_keys[i] = (Ci, Di)
        del round_keys[0]
        #form the keys from concatenated CiDi 1<=i<=16 and by
    apllying PC2
        # WRITE YOUR CODE HERE!
        for i in range(1, 17):
            (Ci, Di) = round_keys[i]
            round_keys[i] = permutation_by_table((Ci << 28) | Di,
    56, PC2)
        return round_keys
```

3. Implementing the f function

F 函数使得 32 位的半分组 Ri 经过 E 置换，相应轮次密钥 Ki 异或，S 盒替换，P 盒置换后，输出 32 位结果。

```
    def round_function(Ri, Ki):
        # expand Ri from 32 to 48 bit using table E
        Ri = permutation_by_table(Ri, 32, E)

        # xor with round key
        # WRITE YOUR CODE HERE!
        Ri = Ri ^ Ki
        # split Ri into 8 groups of 6 bit
        # WRITE YOUR CODE HERE!
        groups = dict.fromkeys(range(0, 8))
        for i in range (0, 8):
            groups[i] = Ri & (2**6 -1)
            Ri = Ri >> 6
        # interpret each block as address for the S-boxes
        # WRITE YOUR CODE HERE!
        S_Values = {}
        for i in range(0, 8):
            group_value = groups[7-i]
            row = (group_value >> 5 << 1 )+ (group_value & 1)
            col = (group_value >> 1) & 2**4-1
            S_Values[i] = Sboxes[i][row*16 + col]
        # pack the blocks together again by concatenating
        # WRITE YOUR CODE HERE!
        Ri = 0
        for i in range(0, 8):
            Ri = (Ri << 4) + S_Values[i]
        # another permutation 32bit -> 32bit
        Ri = permutation_by_table(Ri, 32, P)
        return Ri
```

4. Implementing the encrypt function

由于 DES 加解密的对称性，encrypt 函数中可完成加密和解密两种操作，两者之间的唯一差别在于对 16 个子密钥的使用顺序，所以定义了参数 decrypt，当 decrypt=False 时，顺序使用子密钥，完成加密操作，当 decrypt=True 时，倒叙使用子密钥，完成解密操作。

可在此函数中对 DES 流程有一个整体认识：

1. 密钥处理：对密钥进行 PC1 置换、拆分、生成 round_keys;
2. 数据处理：对分组进行 IP 置换并拆分；
3. 加（解）密过程：使用相应的子密钥顺序对数据进行 16 轮处理；
4. 最终变换：为使得加解密过程一致，连接半分组后对分组进行逆 IP 置换。

```
1       def encrypt(msg, key, decrypt=False):
2           # permutate by table PC1
3           key = permutation_by_table(key, 64, PC1) # 64bit -> PC1 ->
        56bit
4           # split up key in two halves
5           # WRITE YOUR CODE HERE!
6           C0 = key >> 28
7           D0 = key & (2**28-1)
8           #
9           # # generate the 16 round keys
10          round_keys = generate_round_keys(C0, D0) # 56bit -> PC2 ->
        48bit
11          #
12          msg_block = permutation_by_table(msg, 64, IP)
13          # # WRITE YOUR CODE HERE!
14          L0 = msg_block >> 32
15          R0 = msg_block & (2**32-1)
16          if not decrypt:
17              for i in range(0, 16):
18                  Li = R0
19                  Ri = L0 ^ round_function(R0, round_keys[i + 1])
20                  R0 = Ri
21                  L0 = Li
22          else:
23              for i in range(15, -1, -1):
24                  Li = R0
25                  Ri = L0 ^ round_function(R0, round_keys[i + 1])
26                  R0 = Ri
27                  L0 = Li
28          L_last = R0
29          R_last = L0
30          # WRITE YOUR CODE HERE!
31          text = (L_last << 32) + R_last
```

```
32          # final permutation
33          text = permutation_by_table(text, 64, IP_INV)
34          return str(hex(text))[2:]
```

5. Decrypt the DES Ciphertext

使用密钥对密文进行解密，得到相应的明文。通过解密结果可大致推断解密内容是正确的。为了验证加密过程的正确性，并进一步验证解密过程，我们使用 encrypt 对得到的明文进行加密，再与最初给到的密文进行比较。

```
C:\Users\90948\PycharmProjects\DES\venv\Scripts\python.exe
Plain Text: WuhanV5!
加解密可逆！

Process finished with exit code 0
```

Figure 3

```
1       def decrypt(cipher_text, key):
2           plain_text = encrypt(cipher_text, key, decrypt=True)
3           print('Plain Text: '+
        bytearray.fromhex(plain_text).decode())
4           return int(plain_text, 16)
5
6       def encrypt2(cipher_text, key):
7           plain_text = encrypt(cipher_text, key, decrypt=False)
8           return int(plain_text, 16)
9
10      import binascii
11      key = int(binascii.hexlify(b'FudanNiu') , 16)
12      cipher_text_odd = 0x9b99d07d9980305e
13      # cipher_text_even = 0x6f612748df99a70c
14      plain_text = decrypt(0x9b99d07d9980305e, key)
15      cipher_text = encrypt2(plain_text, key)
16      if cipher_text == cipher_text_odd:
17          print("加解密可逆！")
```

6. Decrypt the AES Ciphertext

使用 OPENSSL 以及 DES 解密得到的密钥对‘odd.enc’进行解密，得到明文‘odd.plain’。

```
ahho@ubuntu:~/Desktop$ openssl enc -d -aes256 -in odd.enc -out odd.plain
enter aes-256-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

Figure 4

After screenshots of his WeChat messages were shared on Chinese forums and gained huge attention, the supervision department summoned him to talk, where he was blamed for leaking the information. On 3 January 2020, police from the Wuhan Public Security Bureau investigated the case and interrogated Li, giving him a warning notice and censuring him for "making false comments on the Internet". He was made to sign a letter of admonition promising not to do it again. The police warned him that if he failed to learn from the admonition and continued to violate the law he would be prosecuted.

四、实验结果

根据 DES 解密结果以及 AES 解密后的明文，我们可以判断过程无误。

在实验过程中，完整地完成实现过程仅仅只能针对不超过 64bit 的密钥密文进行加解密，为了完善 DES，使得对于密钥有检验，并能应对长密文，我们还可以略作改进。

同时由于 DES 密钥长度的限制，可以考虑使用 3DES 代替 DES:

1. 用密钥 K1 进行 DEA 加密。
2. 用 K2 对步骤 1 的结果进行 DES 解密。
3. 用步骤 2 的结果使用密钥 K1 进行 DES 加密。

这种方法的缺点，是要花费原来三倍时间，从另一方面来看，3DES 的 112 位密钥长能有效提高加密的健壮性。

五、实验总结

通过“DES&AES”实验，我们可以更加清楚地了解对称分组加密，用实际的代码实现 DES 加解密。可以看出，通过大量的“混乱和扩散”，整个加密过程变得相当复杂，想要直接从原理上找到 DES 的漏洞变得不太可能，相比于维吉尼亚密码有很大优势，所以长久以来，DES 也仅仅是因为密钥长度过短，屈服于穷举攻击之下。