实验报告

【实验目的】

1. 理解 $\mathrm{Fesitel}$结构及 DES 算法的原理，并掌握加解密流程；
2. 了解弱密钥与半弱密钥；
3. 对算法进行深入理解和思考，开始尝试算法的优化与改良；
4. 必做部分 DES 算法的实现需要给出流程图和伪代码。

【实验环境】

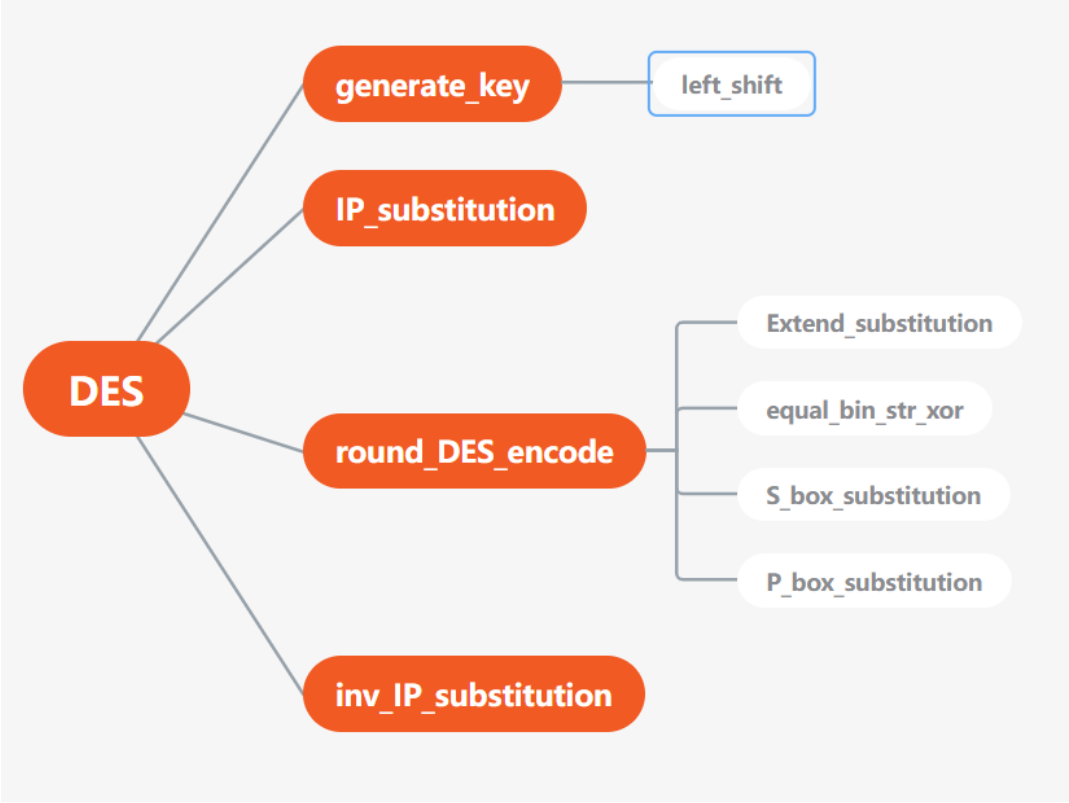
1. 语言：C
2. 平台：clion 2021.2 版本

【实验内容】

一、实现DES算法

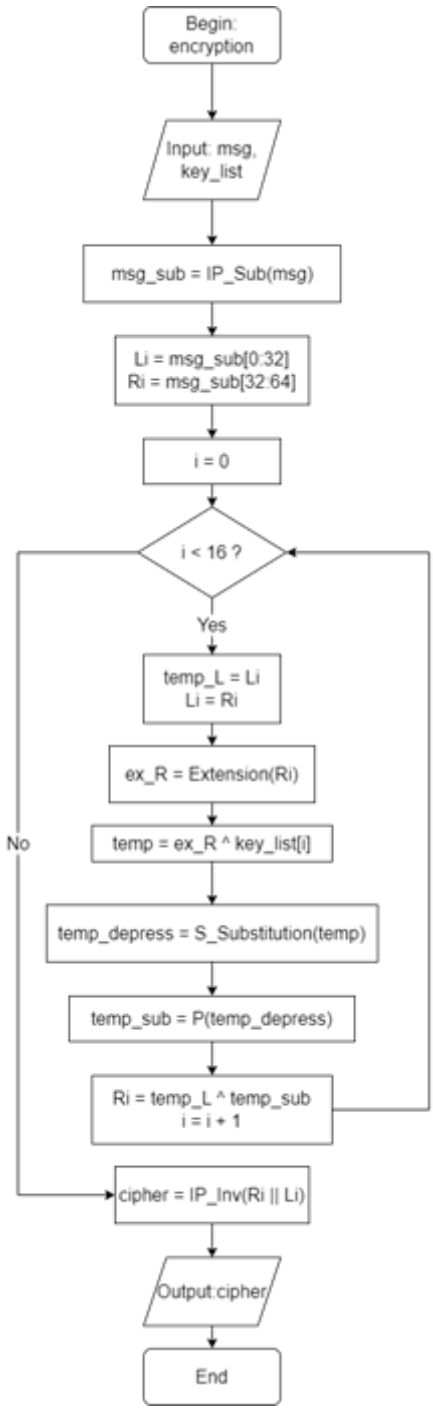
1. **算法流程**

   - **函数调用图如下：**



   - **函数流程图如下：**

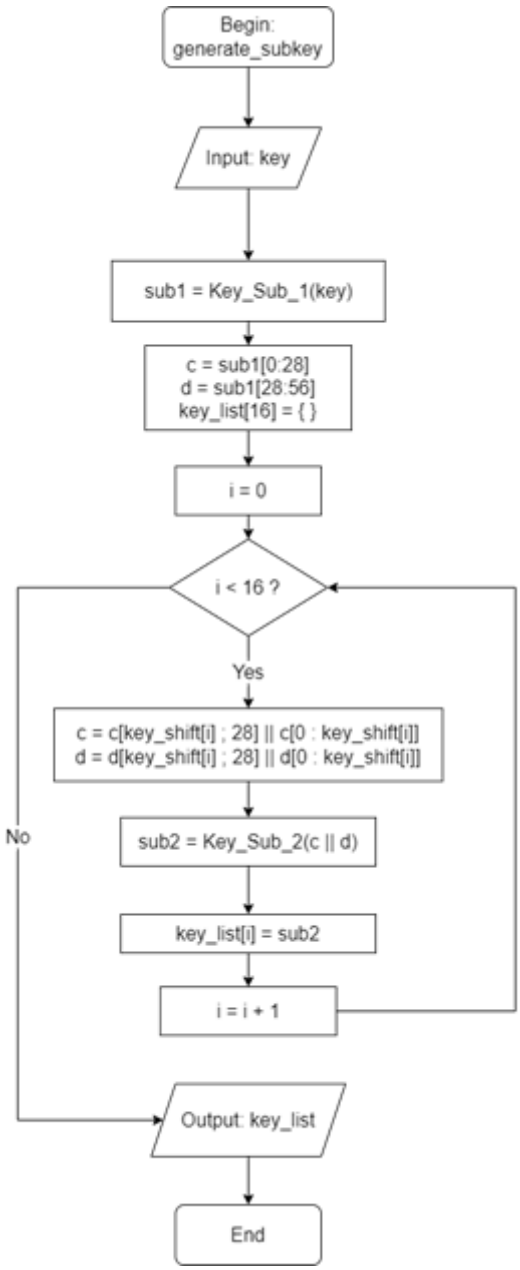
     - 加密：



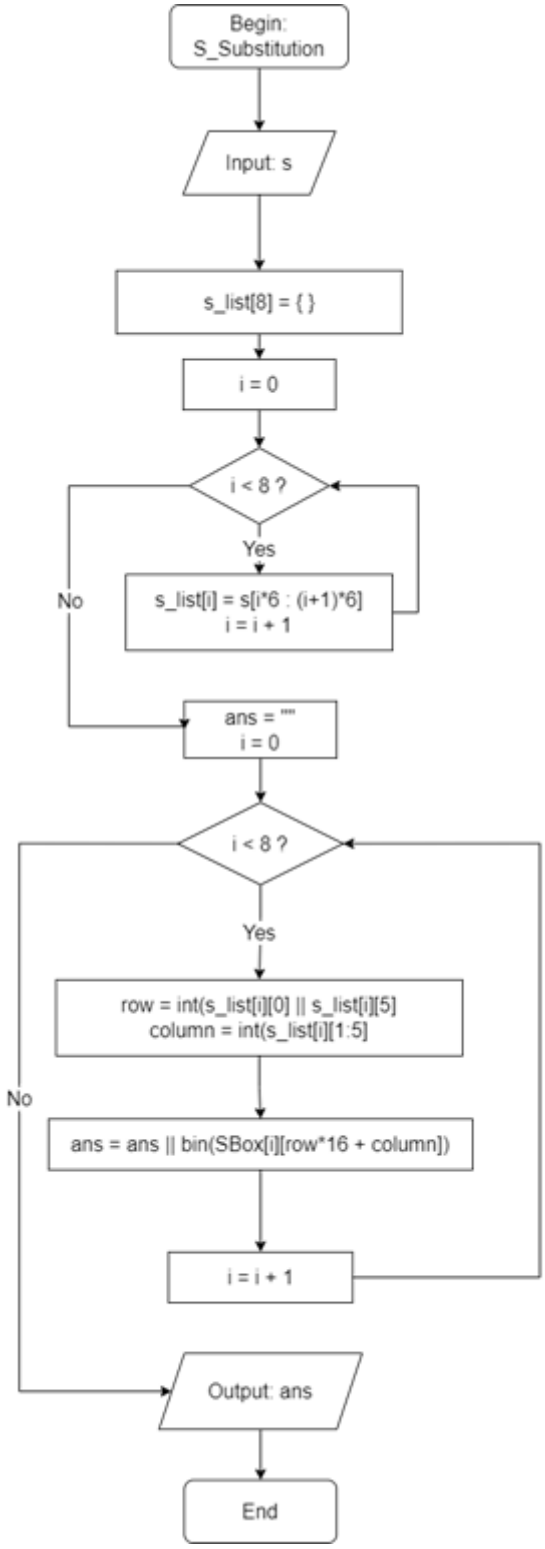
- 解密：

  由于DES采用的是Fesitel结构，对于解密而言仅存在轮密钥使用顺序与加密相反，在这里不再给出解密流程图。

- 生成子密钥list的流程图：


Begin:
generate_subkey
Input: key
sub1 = Key_Sub_1(key)
c = sub1[0:28]
d = sub1[28:56]
key_list[16] = { }
i = 0
i < 16 ?
Yes
No
c = c[key_shift[i] ; 28] || c[0 : key_shift[i]]
d = d[key_shift[i] ; 28] || d[0 : key_shift[i]]
sub2 = Key_Sub_2(c || d)
key_list[i] = sub2
i = i + 1
Output: key_list
End

- S盒替换的流程图：


Begin:
S_Substitution
Input: s
s_list[8] = { }
i = 0
i < 8 ?
Yes
No
s_list[i] = s[i*6 : (i+1)*6]
i = i + 1
ans = ""
i = 0
i < 8 ?
Yes
No
row = int(s_list[i][0] || s_list[i][5]
column = int(s_list[i][1:5]
ans = ans || bin(SBox[i][row*16 + column])
i = i + 1
Output: ans
End

- **伪代码如下：**

表 1 DES 加密算法伪代码

Algorithm1: DES 加密算法

Input: 明文 msg，子密钥列表 key_list

Output: 密文 cipher

```
function encryption(msg, key_list):
    msg = bin(int(msg, 16))[2:]
    msg_sub = IP_Sub(msg)
    Li = msg_sub[0:32]
    Ri = msg_sub[32:64]
    for i = 0 to 16 do
        temp_L = Li
        Li = Ri
        ex_R = Extension(Ri)
        temp = ex_R ^ key_list[i]
        temp_depress = S_Substitution(temp)
        temp_sub = PBox(temp_depress)
        Ri = temp_L ^ temp_sub
    end for
    cipher = IP_Inv(Ri || Li)
    return cipher
```

表 2 DES 解密算法伪代码

Algorithm1: DES 解密算法

Input: 明文 msg，子密钥列表 key_list

Output: 密文 cipher

```
function decryption(cipher, key_list):
    cipher = bin(int(cipher, 16))[2:]
    cipher_sub = IP_Sub(cipher)
    Li = cipher_sub[0:32]
    Ri = cipher_sub[32:64]
    for i = 0 to 16 do
        temp_L = Li
        Li = Ri
        ex_R = Extension(Ri)
        temp = ex_R ^ key_list[15 − i]
        temp_depress = S_Substitution(temp)
        temp_sub = PBox(temp_depress)
        Ri = temp_L ^ temp_sub
    end for
    msg = IP_Inv(Ri || Li)
    return msg
```

- 生成子密钥的伪代码：

表 3 子密钥生成算法伪代码

Algorithm1: DES 子密钥生成算法

Input: 密钥 key

Output: 子密钥列表 key_list

```
function generate_subkey(key):
    sub_1 = Key_Sub_1(key)
    c = sub_1[0:28]
    d = sub_1[28:56]
    key_list[16] = {}
    for i = 0 to 16 do
        c = c[key_shift[i]:28] + c[0:key_shift[i]]
        d = d[key_shift[i]:28] + d[0:key_shift[i]]
        sub_2 = Key_Sub_2(c||d)
        key_list[i] = sub_2
    end for
    return key_list
```

- S盒替换的伪代码：

表 4 S 盒替换伪代码

Algorithm1: DES 的 S 盒替换算法

Input: 48 位字符串 s

Output: 压缩后的 32 位字符串 ans

```
function S_Substitution(s):
    s_list[8][6] = {}
    for i = 0 to 8 do
        s_list[i] = s[i * 6 : (i + 1) * 6]
    end for
    ans = ""
    for i = 0 to 8 do
        row = int(s_list[i][0]||s_list[i][5], 2)
        column = int(s_list[i][1:5], 2)
        ans = ans||bin(SBox[i][row * 16 + column])[2:]
    end for
    return ans
```

2. 测试样例及结果截图：

本地测试样例的运行结果如下所示：

```
1
0x900eb9584f9321c0
0x84f5ca4805eb1ba3
1
0xb144bfab09125c7b
```

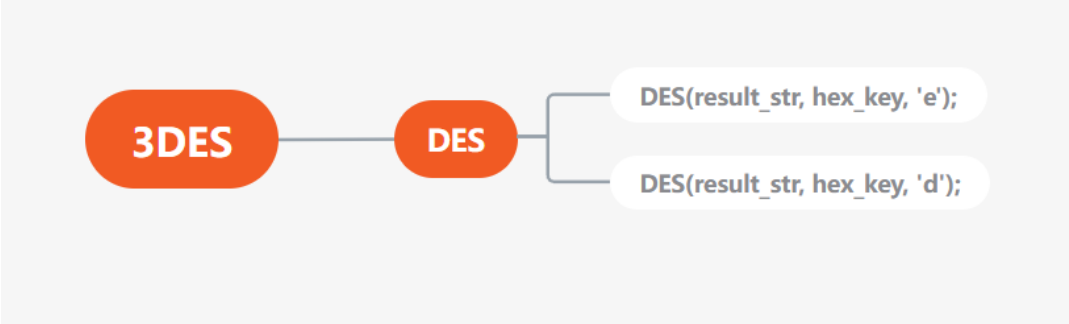
3. **讨论与思考：**

DES的加解密其实不难，但有一个需要注意的地方在于前置0的添加，使其成为标准的位数，否则会导致在加密过程中的运算错误。

二、三重DES算法

1. **算法流程：**

   - **函数调用图：**



2. **测试样例及结果截图：**

```
0x78e2025d31d06fde
0xa530af81d46635e4
0xfc6aa9db1d2b1224
1
0xf1b98dac6657545c
```

```
0x618f1b32353b4f35
0xc50f4b2e21282135
0x35becf7d711ee29a
0
0xe5388d6e1eda0fad
```

3. **讨论与思考：**

当$k_1 = k_2$时，由于DES加解密的对称性，三重DES加密退化为基础的一重DES加密。

三、弱密钥与半弱密钥

算法流程：

1. 函数调用图：

   本题只有一个main函数，未进行其他函数的调用。

2. 具体思路见如下链接：

   Weak key - Wikipedia

   - DES具有一些称为"弱密钥"和"半弱密钥"的特定密钥。这些密钥会导致 DES 的加密模式与 DES 的解密模式（尽管可能是不同密钥的解密模式）相同。
   - 在操作中，秘密56位密钥根据DES密钥计划分为16个子密钥；在十六个 DES 回合中，每个回合使用一个子项。DES弱密钥生成16个相同的子项。当密钥（以16进制表示）时，就会发生这种情况。
     - 交替 1 + 0 （0x0101010101010101）
     - 交替"F"+"E"（0xFEFEFEFEFEFEFEFE）
     - "0xE0E0E0E0F1F1F1F1"
     - "0x1F1F1F1F0E0E0E0E"
   - 如果实现不考虑奇偶校验位，则具有反转奇偶校验位的相应密钥也可以用作弱密钥：
     - 所有零（0x0000000000000000）
     - 所有F（0xFFFFFFFFFFFFFFFF）
     - "0xE1E1E1E1F0F0F0F0"
     - "0x1E1E1E1E0F0F0F0F"
   - 使用弱密钥，DES密钥PC_1置换的结果会导致舍入键全部为零、全 1 或交替的零一模式。

     由于所有子项都是相同的，并且DES是Feistel网络，因此加密功能是自反转的;也就是说，尽管加密一次，给出一个看起来安全的密文，但加密两次会产生原始的明文。
   - DES 还具有半弱密钥，它只生成两个不同的子项，每个子项在算法中使用八次：这意味着它们成对出现 *K1* 和 *K2*，并且它们具有以下属性：

     其中$E_k(m)$是使用密钥 *K* 加密消息*M* 的加密算法。有六个半弱密钥对：
     - 0x011F011F010E010E 和 0x1F011F010E010E01
     - 0x01E001E001F101F1 和 0xE001E001F101F101
     - 0x01FE01FE01FE01FE 和 0xFE01FE01FE01FE01
     - 0x1FE01FE00EF10EF1 和 0xE01FE01FF10EF10E
     - 0x1FFE1FFE0EFE0EFE 和 0xFE1FFE1FFE0EFE0E
     - 0xE0FEE0FEF1FEF1FE 和 0xFEE0FEE0FEF1FEF1

     还有 48 个可能的弱键只生成四个不同的子项（而不是 16 个）。
   - 弱密钥的生成方法较为简单：
     1. 首先确定压缩分组后的c和d的形式：
     2. 由于循环左移和置换算法，子密钥若相等，则需要保证进行压缩置换2前的c_i和d_i也相等。第一次循环左移位数为1，这一步就使得：左移前和左移后若想等的话，则每一位必须全部相同，也即全为1或者全为0。
     3. 因此c_0和d_0的备选项即为28位的全0串和全1串，共4种组合。
     4. 组合后用置换1逆置换后，再补充奇偶校验位即可生成全部的弱密钥。
   - 半弱密钥的生成：
     1. 半弱密钥的c_0和d_0的备选项也包含28位的全0串和全1串，但不能二者同时全0或全1，否则就是弱密钥，半弱密钥首先需要保证这对密钥是不相等的，所以还需要考察别的备选项。
     2. 不妨假设一个密钥c_0的前两位是“10”，其余位待定，再待定另一个密钥的c_0是未知的。

3. 循环左移16次时相当于循环左移了28位，也即与原来的保持不变，再让第一个密钥c_0循环左移一位，另二者相等，即可确定另一密钥的c_0其中两位。
4. 随后，按照循环左移表，依次让第一个密钥的c_0循环左移，让另一密钥的c_0循环右移，不断令二者相等，即可陆续确定其余未知位的值。
5. 最终构造出“10”交替串和“01”交替串是一对半弱密钥c_0和d_0的备选项。
6. 保证c_0和d_0至少有其一为“10/01”交替串，共6种组合，后续步骤与弱密钥类似，即可构造出12对带奇偶校验的半弱密钥组合。

测试样例及结果截图：

```
0x010101010101010101
0x000000000000000000
0x1f1f1f1f0e0e0e0e0e
0x1e1e1e1e0f0f0f0f0f
0xe0e0e0e0f1f1f1f1f1
0xe1e1e1e1f0f0f0f0f0
0xfefefefefefefefefe
0xffffffffffffffffff
0x1f011f010e010e01 0x011f011f010e010e
0xfee0fee0fef1fef1 0xe0fee0fef1fef1fe
0x01e001e001f101f1 0xe001e001f101f101
0x1ffe1ffe0efe0efe 0xfe1ffe1ffe0efe0e
0x01fe01fe01fe01fe 0xfe01fe01fe01fe01
0xe01fe01ff10ef10e 0x1fe01fe00ef10ef1
0x1e001e000f000f00 0x001e001e000f000f
0xffe1ffe1fff0fff0 0xe1ffe1fff0fff0ff
0x00e100e100f000f0 0xe100e100f000f000
0x1eff1eff0fff0fff 0xff1eff1eff0fff0f
0x00ff00ff00ff00ff 0xff00ff00ff00ff00
0xe11ee11ef00ff00f 0x1ee11ee10ff00ff0
```

讨论与思考：

1. 由于DES的加密解密算法是对称的。弱密钥和半弱密钥的安全风险均很大。
2. 弱密钥的危害在于，用其生成的密文再用该弱密钥加密一次即可得到明文。
3. 半弱密钥的危害在于，用一对半弱密钥中的其一加密后，再用与之对应的半弱密钥加密后即可得到明文。

四、DES优化：

1. **由于我从始至终都是拿c写代码，所以这块其实并没有优化...**
2. 前段时间在搞AES，所以没来得及对算法上进一步的优化。
3. **思考题：**

   目前对于DES密码有差分密码分析攻击和线性密码分析攻击两种攻击方式，尽管能加速DES的破解，但相比穷举攻击的提升比例有限。DES的安全性依赖于密钥。如今电脑的计算性能已经远比过去要强，而DES的密钥有效位只有56位，不适用于现代的主要原因就在于此，它已经不足以抵抗高性能计算机的攻击。(只能说不如AES)

   通过本次实验，掌握了DES这一较为基础的加解密算法的实现原理，也体会到了混淆和扩散这两大原则在密码中的作用。同时也了解了密钥对于此类对称加密算法的重要性，认识到了弱密钥/半弱密钥的危害程度。其中所蕴含的思想也会在后续的加密算法中一一体现，希望之后的实验我能在此基础上更好地完成。