DES 算法的程序设计和实现——实验报告

一、算法原理概述

DES 是一种典型的块加密方法。它以 64 位为分组长度，64 位一组的明文作为算法的输入，通过一系列复杂的操作，输出同样 64 位长度的密文。DES 使用加密密钥定义变换过程，因此算法认为只有持有加密所用的密钥的用户才能解密密文。DES 采用 64 位密钥，但由于每 8 位中的最后 1 位用于奇偶校验，实际有效密钥长度为 56 位。密钥可以是任意的 56 位的数，且可随时改变。DES 算法的基本过程是换位和置换。

加密过程：

$$C = E_k(M) = IP^{-1} \cdot W \cdot T_{16} \cdot T_{15} \cdots \cdot T_1 \cdot IP(M).$$

解密过程：

$$M = D_k(C) = IP^{-1} \cdot W \cdot T_1 \cdot T_2 \cdot \cdots \cdot T_{16} \cdot IP\ (C)\ .$$

变量说明：

①M 为算法输入的 64 位明文块；

②Ek 描述以 K 为密钥的加密函数，由连续的过程复合构成；

③IP 为 64 位初始置换；

④$T_1, T_2, \cdots, T_{16}$ 是一系列的迭代变换；

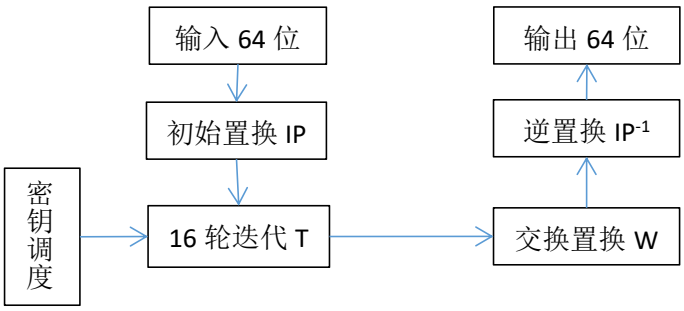
⑤W 为 64 位置换，将输入的高 32 位和低 32 位交换后输出；

⑥IP^{-1} 是 IP 的逆置换；

⑦C 为算法输出的 64 位密文块

二、算法总体结构

DES 算法的总体结构为 Feistel 结构。



三、模块分解

DES 算法模块分为两个大模块，一个是子密钥产生，一个是加解密过程。

子密钥产生部分分为 3 个模块：

①**PC-1 置换**：对 K 的 56 个非校验位实行置换 PC-1，得到 C_0D_0，其中 C_0 和 D_0 分别由 PC-1 置换后的前 28 位和后 28 位组成。

②**16 次 LS 置换**：计算 $C_i = LS_i(C_i\text{-}1)$ 和 $D_i = LS_i(D_i\text{-}1)$。当 i=1, 2, 9, 16 时，$LS_i(A)$表示将二进制串 A 循环左移一个位置；否则循环左移两个位置。

③**PC-2 压缩置换**：对 56 位的 C_iD_i 实行 PC-2 压缩置换，得到 48 位的 K_i

加密和解密过程分为 6 个模块：

①**获取子密钥**：使用子密钥产生模块

②**原始明文消息按 PKCS#5 (RFC 8018)规范进行字节填充**：原始明文消息最后的分组不够 8 个字节 (64 位) 时，在末尾以字节填满，填入的字节取值相同，都是填充的字节数目；原始

明文消息刚好分组完全时，在末尾填充 8 个字节 (即增加一个完整分组)，字节取值都是 08。
③**初始置换 IP**：给定 64 位明文块 M，通过一个固定的初始置换 IP 来重排 M 中的二进制位，得到二进制串 $M_0 = IP(M) = L_0R_0$，这里 L_0 和 R_0 分别是 M_0 的前 32 位和后 32 位。
④**16 轮迭代 T**: 根据 L_0R_0 按下述规则进行 16 次迭代, 即 $L_i = R_{i-1}$, $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$, i = 1 .. 16。输入 64 位明文 M 时，子密钥按 ($K_1K_2 \cdots K_{16}$)的次序调度，是加密过程。输入 64 位密文 C 时子密钥按 ($K_{16}K_{15} \cdots K_1$)的次序调度。
⑤**交换置换 W**：16 次迭代后得到 $L_{16}R_{16}$；左右交换输出 $R_{16}L_{16}$。
⑥**逆置换 IP^{-1}**：对迭代 T 输出的二进制串 $R_{16}L_{16}$ 使用初始置换的逆置换 IP_{-1} 得到密文 C。
注：实际编程过程中为了编程方便，实际编写的函数模块可能与上述模块的分类有所出入，但总体结构和算法步骤与上述 DES 算法相同。

四、数据结构

本程序设计使用 C++ STL 中的 string 作为存储数据结构，利用一维和二维数组保存 DES 算法所需要的置换矩阵，用于进行明文和密文的比特位的置换。

五、编译运行结果

注：由于 C 语言中一个 char 为一个字节（8 个 bit）。因此为了测试方便，64 位明文我使用 8 个 char 来表示，密钥同样使用 8 个 char 来表示，但密文使用 64 位二进制数表示。
每个测例中含有下面两个方面的测试（将代码宏定义 DEBUG 设为 1 可观察每一步的结果）：
1.DES 加密测试
输入：明文字符串、密钥
输出：密文字符串
2.DES 解密测试
输入：密文字符串、密钥
输出：明文字符串
测试顺序：将明文输入，得到密文；再将密文输入，得到明文。比较前后两段明文是否一致。
①明文不足 8 个字节

```
Plaintext: abcd
Key: 11111111
result: 1101111101011111101110100101001111000000010110101000101100001110

Ciphertext: 1101111101011111101110100101001111000000010110101000101100001110
Key: 11111111
result: abcd
```

②明文为 8 个字节

```
Plaintext: abcdefgh
Key: 12345678
result: 1100100011101100111010011101010011100110001001110011011100101000101010011001001111000001111010100111011010111011
0011110000100110

Ciphertext: 1100100011101100111010011101010011100110001001110011011100101000101010011001001111000001111010100111011010111
1011001111000010011O
Key: 12345678
result: abcdefgh
```

③明文大于 8 个字节

```
Plaintext: abcdefghijk
Key: 12345678
result: 1100100011101100111010011101010011100110001001110011011100101000100111010011000100101110001100111111100001111101
1111111101101101

Ciphertext: 1100100011101100111010011101010011100110001001110011011100101000100111010011000100101110001100111111100001111
1101111111101101101
Key: 12345678
result: abcdefghijk
```