实验指导书 3

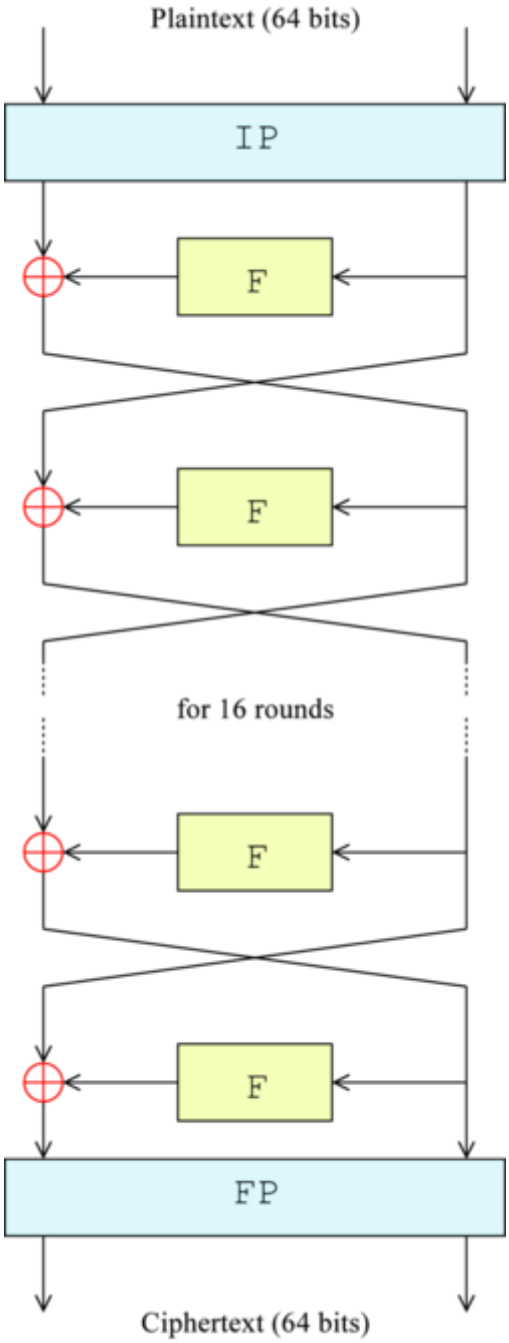
DES算法属于分组加密算法，即在明文加密和密文解密过程中，信息都是按照固定长度分组后进行处理的。混淆和扩散是它采用的两个最重要的安全特性。混淆是指通过密码算法使明文和密文以及密钥的关系非常复杂，无法从数学上描述或者统计。扩散是指明文和密钥中的每一位信息的变动，都会影响到密文中许多位信息的变动，从而隐藏统计上的特性，增加密码的安全。

DES算法将明文分成64位大小的众多数据块，即分组长度为64位。同时用56位密钥对64位明文信息加密，最终形成64位的密文。

需要注意的地方是掌握DES算法的16轮加、解密流程以及子密钥的产生流程。

DES算法16轮加、解密流程

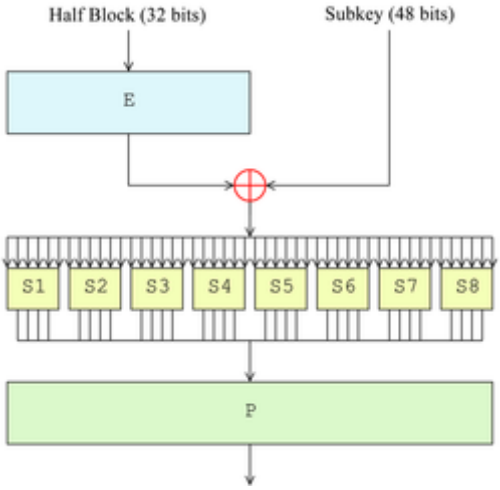
加密的整个流程图如下：



首先是将明文(64 bits)进行置换，即上图中IP环节，其中FP表示IP的逆置换。

然后将置换后的结果分为两个32位，左边记为L0，右边记为R0。可以从上面的图看见Ri+1为Ri经过函数F处理后和L0异或的结果。这里会用到子密钥Ki，因为后文会介绍子密钥产生流程，这里就不赘述。

在F函数里面，进行的变化如下图：



n 图中的E表示扩展，会将原本的32位扩展位48位，即先按照每行4个bit排列，然后在每行的首位(记为i)插入i-1位，如果是1则插入第32位，在每行行尾(记为j)插入j+1位，如果是第32位则插入第1位。从而扩展成48位。

n 然后和子密钥进行异或。

n 下面的进行S盒变换，每六位输入，输出为4位。六位的输入中首位和尾位作为S盒行数，中间四位作为S盒的列数。

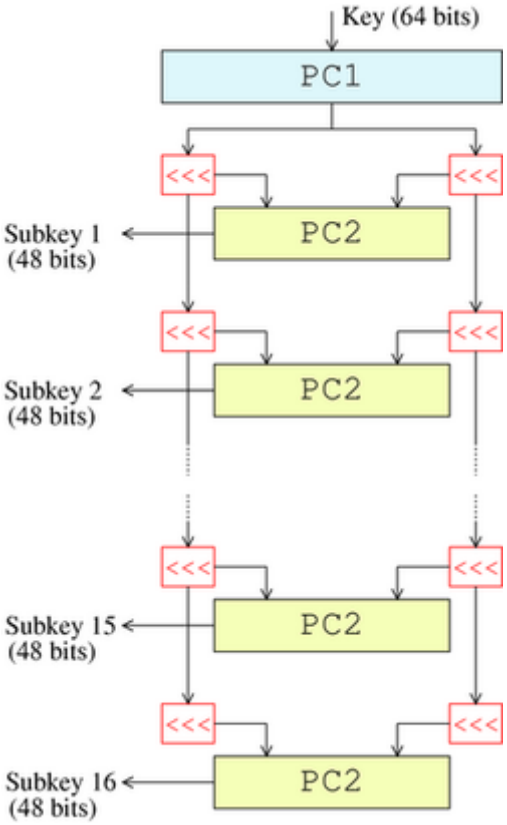
n 最后的P是一个置换函数。

最后的FP，前文已经说过是IP的一个逆变换，将输入的Li和Ri进行置换拼接即可。

至此整个加密流程结束。DES的解密流程即与DES加密流程刚好相反（F和E这些函数不会变化），便不必再赘述。

子密钥的产生以及弱密钥和半弱密钥的验证

子密钥产生的流程图如下：



首先是将密钥(64 bits)进行置换处理，PC1和PC2均表示置换流程。其中PC1规模为8*7，因此进行PC1置换后的变为了56 bits。

将这56 bits分为两半，进行循环左移（记第i轮循环左移为LSi），其中LS1、LS2、LS9、LS16进行循环左移一位，其余的LSi都循环移位2位。

循环移位后将这两个28 bits进行拼接，并根据PC2进行置换，PC2的规模为8*6。因此置换后产生的子密钥均为48 bits。

对于弱密钥和半弱密钥的验证，采用c++验证，调用cryptopp库进行DES加密。因为弱密钥是指进行两次加密就会加密回原文，半弱密钥对指先后采用这两个密钥加密也会加密会原文，按照这个逻辑进行程序书写。

```
//弱密钥
unsigned char weakKetSets[4][DES::DEFAULT_KEYLENGTH] = {
        {0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01},
        {0x1F,0x1F,0x1F,0x1F,0x0F,0x0F,0x0E,0x0E},
        {0xE0,0xE0,0xE0,0xE0,0xF1,0xF1,0xF1,0xF1},
        {0xFE,0xFE,0xFE,0xFE,0xFE,0xFE,0xFE,0xFE}
};
//半弱密钥对
unsigned char halfWeakKey[4][DES::DEFAULT_KEYLENGTH]{
    {0x01,0xFE,0x01,0xFE,0x01,0xFE,0x01,0xFE},
    {0xFE,0x01,0xFE,0x01,0xFE,0x01,0xFE,0x01},
    {0x01,0xE0,0x01,0xE0,0x01,0xF1,0x01,0xF1},
    {0xE0,0x01,0xE0,0x01,0xF1,0x01,0xF1,0x01}
};
//格式化打印结果
void show(unsigned char key[DES::DEFAULT_KEYLENGTH]) {
    for (int i = 0; i < DES::DEFAULT_KEYLENGTH; i++) {
        cout << hex<<setw(2)<<setfill('0')<< (int)key[i] << " ";
    }
}

//弱密钥测试
void WeakTest() {
    //主要是打印一些基本信息，方便调试：
    cout << "DES Parameters: " << endl;
    cout << "Algorithm name : " << DES::StaticAlgorithmName() << endl;

    unsigned char input[DES::BLOCKSIZE] = "12345";//加密的明文
    unsigned char output[DES::BLOCKSIZE];
    unsigned char output1[DES::BLOCKSIZE];

    cout << "input is: " << input << endl;
    DESEncryption encryption_DES;
    cout << "----弱密钥测试----" << endl;
    //密码测试
    for (int i = 0; i < 4; i++) {
        encryption_DES.SetKey(weakKetSets[i], DES::KEYLENGTH);
        encryption_DES.ProcessBlock(input, output);
        encryption_DES.ProcessBlock(output, output1);
       //比对加密的密文是否和明文一样，后面都是这个目的
        if (memcmp(output1, "12345", 5) == 0) {
            show(weakKetSets[i]);
            cout << "-->" << "YES" << endl;
        }
        else {
            show(weakKetSets[i]);
            cout << "-->" << "NO" << endl;
        }
    }

    cout << "----半弱密钥测试----" << endl;
    for (int i = 0; i < 4; i += 2) {
        encryption_DES.SetKey(halfWeakKey[i], DES::KEYLENGTH);
        encryption_DES.ProcessBlock(input, output);
        encryption_DES.SetKey(halfWeakKey[i + 1], DES::KEYLENGTH);
        encryption_DES.ProcessBlock(output, output1);
        if (memcmp(output1, "12345", 5) == 0) {
```

```
            show(weakKetSets[i]);
            cout << endl;
            show(weakKetSets[i + 1]);
            cout << "-->" << "YES" << endl;
        }
        else {
            show(weakKetSets[i]);
            cout << endl;
            show(weakKetSets[i + 1]);
            cout << "-->" << "NO" << endl;
        }
    }
    return;
}
```

对文件加密解密的源码

```
void DES_encrypt(unsigned char key[DES::DEFAULT_KEYLENGTH],const string&
file_in,const string& file_out) {
    ECB_Mode<DES>::Encryption cipher{};
    cipher.SetKey(key,DES::DEFAULT_KEYLENGTH);
    ifstream in(file_in, ios::binary);
    ofstream out(file_out, ios::binary);
    FileSource(in,true,new StreamTransformationFilter(cipher,new
FileSink(out)));
    return;
}

void DES_decrypt(unsigned char key[DES::DEFAULT_KEYLENGTH], const string&
file_in, const string& file_out) {
    ECB_Mode<DES>::Decryption cipher{};
    cipher.SetKey(key, DES::DEFAULT_KEYLENGTH);
    ifstream in(file_in, ios::binary);
    ofstream out(file_out, ios::binary);
    FileSource(in, true, new StreamTransformationFilter(cipher, new
FileSink(out)));
    return;
}

//文件加密测试
void test() {
    char path_in[100] = "F:\\原F盘\\密码学\\exp3\\in.txt";
    char path_out_en[100] = "F:\\原F盘\\密码学\\exp3\\out_en.txt";
    char path_out_de[100] = "F:\\原F盘\\密码学\\exp3\\out_de.txt";
    DES_encrypt(weakKetSets[0], path_in, path_out_en);//文件加密
    DES_decrypt(weakKetSets[0], path_out_en, path_out_de);//文件解密
    return;
}
```

十六进制数加密：

这里输入一个BLOCK即64bits的文字，加密完后再修改其中一位，再进行加密，查看加密结果。由于只是查看加密一次后的变化，所以密钥选择之前声明的一个弱密钥。

```
//将信息打印出来
void showbits(unsigned char arr[DES::BLOCKSIZE]) {
```

```
    for (int i = 0; i < DES::BLOCKSIZE;i++) {
        cout << bitset<8>(arr[i]) << endl;
    }
    cout << "-----------" << endl;
    return;
}
//16进制加密
void test2() {
    unsigned char plainText[DES::BLOCKSIZE] = {
0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10};//原文
    cout << "plain text:" << endl;
    showbits(plainText);
    unsigned char output[DES::BLOCKSIZE];
    unsigned char output1[DES::BLOCKSIZE];
    ECB_Mode<DES>::Encryption cipher{};
    cipher.SetKey(weakKetSets[0], DES::DEFAULT_KEYLENGTH);
    cipher.ProcessLastBlock(output, 8, plainText, 8);//加密
    cout << "encryt" << endl;
    showbits(output);
    plainText[7] = 0x12;//修改其中一个bit
    cout << "one bit has been changed:" << endl;
    showbits(plainText);//打印修改后的原文
    cipher.ProcessLastBlock(output1, 8, plainText, 8);
    cout << "encrypt" << endl;
    showbits(output1); //打印加密修改后加密的原文
    return;
}
```