Web Security Assignment 1

数据科学与计算机学院 软件工程专业 向泳郦 16340249

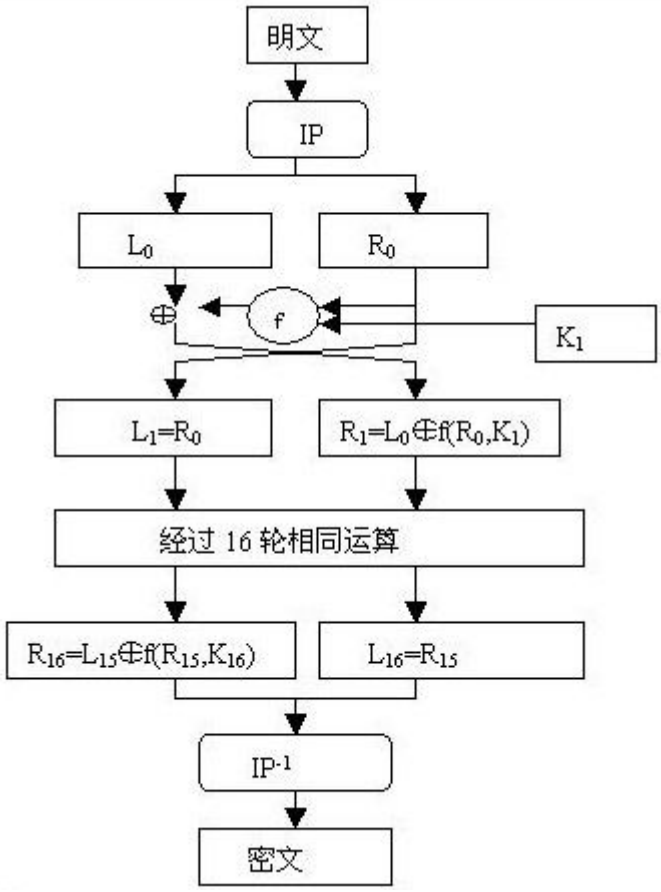
实验需求

- 完成一个 DES 算法的程序设计和实现，包括
  - ✧ 算法原理概述；总体结构；模块分解；数据结构；C 语言源代码；编译运行结果。
  - ✧ No copies of others' are accepted.

算法原理

DES算法为密码体制中的对称密码体制，又被称为美国数据加密标准，是1972年美国IBM公司研制的对称密码体制加密算法。 明文按64位进行分组，密钥长64位，密钥事实上是56位参与DES运算（第8、16、24、32、40、48、56、64位是校验位， 使得每个密钥都有奇数个1）分组后的明文组和56位的密钥按位替代或交换的方法形成密文组的加密方法。

总体结构



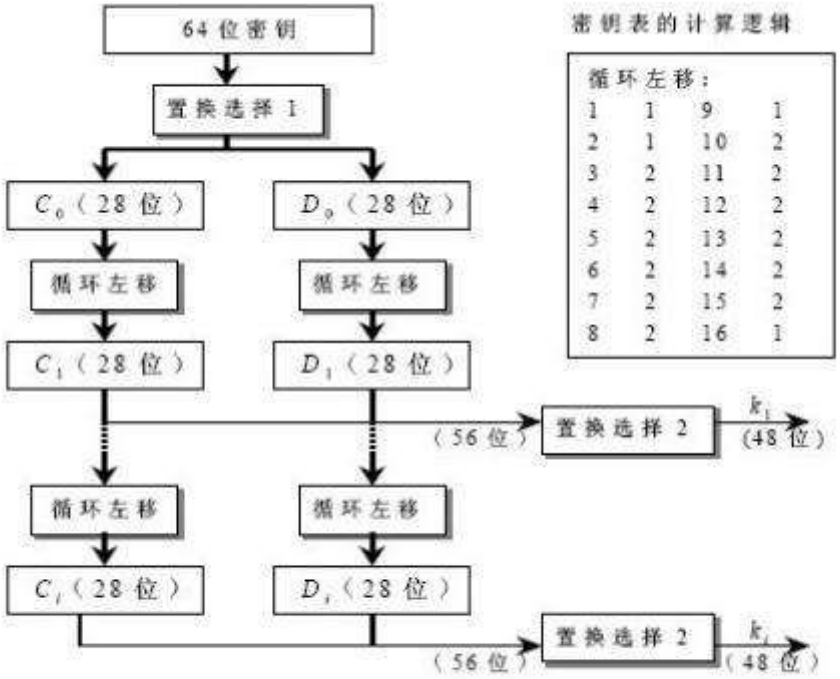
1. 求子钥K1–K16
   K –> 置换得K+ –> 得C0，D0 –> 左移求C1D1–C16D16 –> 置换得K1–K16
2. 利用子钥求L16R16
   (1)M–> 置换得M+ –> 得L0，R0

   (2)进入16次循环：
   Rn拓展置换（32->48）–> 结果异或Kn+1 –> 异或后结果进入S盒（48->32）–> S盒输出进行P置换 –> 结果异或Ln –> 结果赋给Rn+1，Ln+1 = Rn

   (3)得到L1R1–L16R16
3. 生成密文
   L16R16 –> R16L16 –>位置交换表置换得密文
4. 解密过程
   解密过程基本与加密过程相同，不过，加密利用子钥的顺序为K1—K16，解密过程利用子钥的顺序为K16—K1。

模块分析

求子钥K1–K16



4子密钥的生成过程

1. 将密钥KEY(64位)转换成8*8矩阵便于后续计算
2. 舍弃奇偶校验位得8*7矩阵K
3. 通过8*7的交换规则表(perm_key),输入矩阵K，输出结果K+
4. 将K+(8*7)，上下分解为C0,D0(4*7)两个矩阵
5. 根据下表，在每一轮Cn,Dn被赋值为Cn-1左移x位后的结果。重复16次得C1D1 – C16D16

轮数	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
位数	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

6. Kn = CnDn(n = 1,2,……,16)
7. 通过8*6的交换规则表(perm_K),输入矩阵Kn(n = 1,2,……,16)，输出结果Kn+(n = 1,2,……,16)
8. 子钥为Kn+(n = 1,2,……,16)

求L16R16

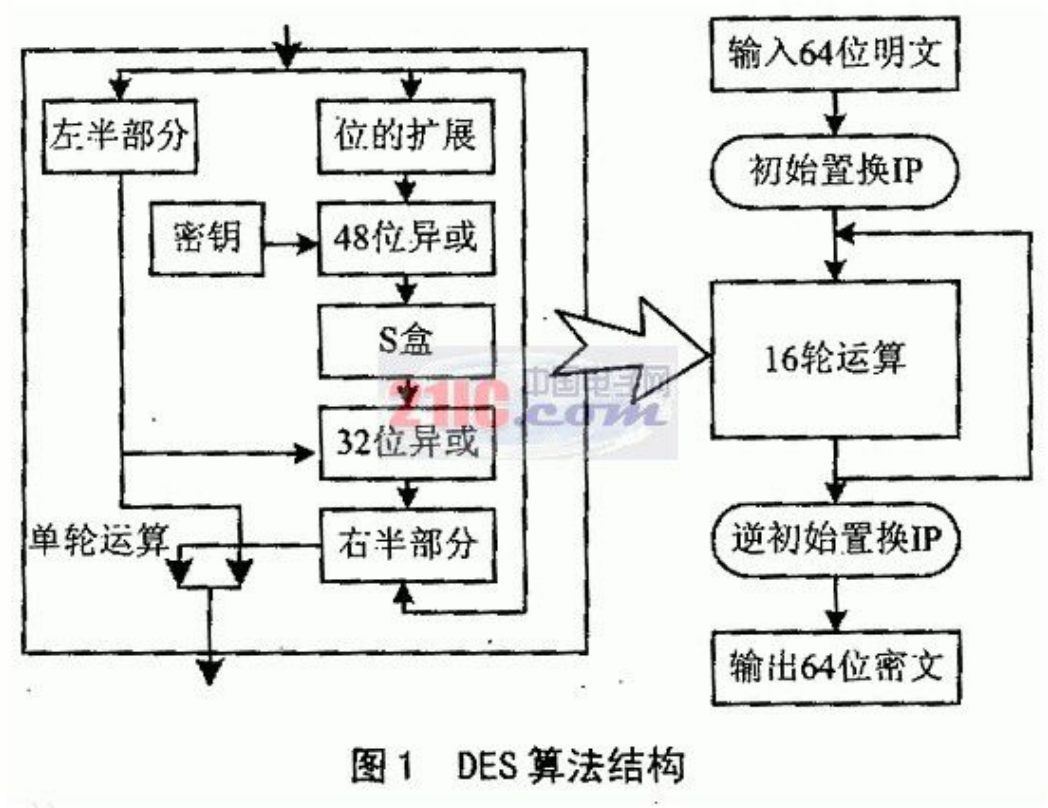


图1 DES算法结构

1. 将密钥M(64位)转换成8*8矩阵便于后续计算
2. 通过8*8的交换规则表(perm_ip),输入矩阵M，输出结果M+
3. 将M+(8*8)，上下分解为L0,R0(4*8)两个矩阵
4. 进入循环，循环次数为16,(n = 1,2,……,16)：
   (1)通过6*8的交换规则表(perm_R),输入矩阵Rn(8*4)，输出结果Rn+(8*6)
   (2)Rn+ = R+ 异或 Kn+1(子钥)
   (3)通过4*16的S盒(perm_S),输入矩阵Rn+(8*6)，输出结果Rn+(8*4)。
   S盒具体规则如下：
   ① 对于矩阵Rn+(8*6),每一行6个为一组，共8组。
   ② 针对每组的6位数，每组中第一位和最后一位组成的二进制代表S盒行数，中间4位组成的数代表列数。
   ③根据行数列数，对应相应组数的S盒中的十进制数
   ④将十进制数转为4位二进制数，替代该组中原6位二进制数
   (4)通过8*4的P盒(perm_P),输入矩阵Rn+，输出结果Rn+
   (5)Rn+ = Rn+ 异或Ln
   (6)Rn+1 = Rn+ Ln+1 = Rn
5. 经过16次循环得，L16、R16

生成密文

1. 将R16(8*4)、L16(8*4)，合并成LR(8*8)

2. 通过8*8的交换规则表(perm_LR),输入矩阵LR，输出结果LR+
3. 将LR+(8*8).转换成16进制数(long long)，得到密文

数据结构

1. 所有密码变化均用二维数组存储，方便变化计算

```
long long KEY = 0x133457799BBCDFF1;
long long M   = 0x7CA66C6EEA3AF51F;

int key_set[8][8] = {0};
int key_set_plus[8][7] = {0};

int m_set[8][8] = {0};
int m_set_plus[8][8] = {0};

int C[17][28] = {0};
int D[17][28] = {0};

int L[17][8][4] = {0};
int R[17][8][4] = {0};

int L_plus[17][8][6] = {0};
int R_plus[17][8][6] = {0};
int R_plus_temp[17][8][4] = {0};

int LR_plus[17][8][8] = {0};
int LR_plus_temp[17][8][8] = {0};

int K[17][8][7] = {0};
int K_plus[17][8][7] = {0};
```

2. 所有规则交换表，P盒，S盒均存储为二维数组，与步骤1统一方便计算

```
int perm_key[8][7] = {
                        {57, 49, 41, 33, 25, 17,  9},
                        {1,  58, 50, 42, 34, 26, 18},
                        {10, 2,  59, 51, 43, 35, 27},
                        {19, 11, 3,  60, 52, 44, 36},
                        {63, 55, 47, 39, 31, 23, 15},
                        {7,  62, 54, 46, 38, 30, 22},
                        {14, 6,  61, 53, 45, 37, 29},
                        {21, 13, 5,  28, 20, 12,  4}
                    };

int num_shift[16] = {1, 1, 2, 2, 2, 2, 2, 2, 1, 2,  2,  2,  2,  2,  2,  1};


int perm_K[8][6] = {
                         {14, 17, 11, 24, 1,   5},
                         {3,  28, 15, 6,  21, 10},
                         {23, 19, 12, 4,  26,  8},
                         {16, 7,  27, 20, 13,  2},
                         {41, 52, 31, 37, 47, 55},
                         {30, 40, 51, 45, 33, 48},
                         {44, 49, 39, 56, 34, 53},
                         {46, 42, 50, 36, 29, 32}
                    };
```

‘’

```
int perm_R[8][6] = {
                        {32, 1,  2,  3,  4,  5},
                        {4,  5,  6,  7,  8,  9},
                        {8,  9,  10, 11, 12, 13},
                        {12, 13, 14, 15, 16, 17},
                        {16, 17, 18, 19, 20, 21},
                        {20, 21, 22, 23, 24, 25},
                        {24, 25, 26, 27, 28, 29},
                        {28, 29, 30, 31, 32, 1}
                    };

int S1[] = {14,  4, 13,  1,  2, 15, 11,  8,  3, 10,  6, 12,  5,  9,  0,  7,
             0, 15,  7,  4, 14,  2, 13,  1, 10,  6, 12, 11,  9,  5,  3,  8,
             4,  1, 14,  8, 13,  6,  2, 11, 15, 12,  9,  7,  3, 10,  5,  0,
            15, 12,  8,  2,  4,  9,  1,  7,  5, 11,  3, 14, 10,  0,  6, 13};

int S2[] = {15,  1,  8, 14,  6, 11,  3,  4,  9,  7,  2, 13, 12,  0,  5, 10,
             3, 13,  4,  7, 15,  2,  8, 14, 12,  0,  1, 10,  6,  9, 11,  5,
             0, 14,  7, 11, 10,  4, 13,  1,  5,  8, 12,  6,  9,  3,  2, 15,
            13,  8, 10,  1,  3, 15,  4,  2, 11,  6,  7, 12,  0,  5, 14,  9};

int S3[] = {10,  0,  9, 14,  6,  3, 15,  5,  1, 13, 12,  7, 11,  4,  2,  8,
            13,  7,  0,  9,  3,  4,  6, 10,  2,  8,  5, 14, 12, 11, 15,  1,
            13,  6,  4,  9,  8, 15,  3,  0, 11,  1,  2, 12,  5, 10, 14,  7,
             1, 10, 13,  0,  6,  9,  8,  7,  4, 15, 14,  3, 11,  5,  2, 12};

int S4[] = { 7, 13, 14,  3,  0,  6,  9, 10,  1,  2,  8,  5, 11, 12,  4, 15,
            13,  8, 11,  5,  6, 15,  0,  3,  4,  7,  2, 12,  1, 10, 14,  9,
            10,  6,  9,  0, 12, 11,  7, 13, 15,  1,  3, 14,  5,  2,  8,  4,
             3, 15,  0,  6, 10,  1, 13,  8,  9,  4,  5, 11, 12,  7,  2, 14};

int S5[] = { 2, 12,  4,  1,  7, 10, 11,  6,  8,  5,  3, 15, 13,  0, 14,  9,
            14, 11,  2, 12,  4,  7, 13,  1,  5,  0, 15, 10,  3,  9,  8,  6,
             4,  2,  1, 11, 10, 13,  7,  8, 15,  9, 12,  5,  6,  3,  0, 14,
            11,  8, 12,  7,  1, 14,  2, 13,  6, 15,  0,  9, 10,  4,  5,  3};

int S6[] = {12,  1, 10, 15,  9,  2,  6,  8,  0, 13,  3,  4, 14,  7,  5, 11,
            10, 15,  4,  2,  7, 12,  9,  5,  6,  1, 13, 14,  0, 11,  3,  8,
             9, 14, 15,  5,  2,  8, 12,  3,  7,  0,  4, 10,  1, 13, 11,  6,
             4,  3,  2, 12,  9,  5, 15, 10, 11, 14,  1,  7,  6,  0,  8, 13};

int S7[] = { 4, 11,  2, 14, 15,  0,  8, 13,  3, 12,  9,  7,  5, 10,  6,  1,
```

‘’

```
int S8[] = {13,  2,  8,  4,  6, 15, 11,  1, 10,  9,  3, 14,  5,  0, 12,  7,
            1, 15, 13,  8, 10,  3,  7,  4, 12,  5,  6, 11,  0, 14,  9,  2,
            7, 11,  4,  1,  9, 12, 14,  2,  0,  6, 10, 13, 15,  3,  5,  8,
            2,  1, 14,  7,  4, 10,  8, 13, 15, 12,  9,  0,  3,  5,  6, 11};

int perm_P[8][4] = {
                        {16,  7,  20,  21},
                        {29,  12,  28,  17},
                        {1,  15,  23,  26},
                        {5,  18,  31,  10},
                        {2,  8,  24,  14},
                        {32,  27,  3,  9},
                        {19,  13,  30,  6},
                        {22,  11,  4,  25}
                    };

int perm_LR[8][8] = {
                        {40,  8, 48, 16, 56, 24, 64, 32},
                        {39,  7, 47, 15, 55, 23, 63, 31},
                        {38,  6, 46, 14, 54, 22, 62, 30},
                        {37,  5, 45, 13, 53, 21, 61, 29},
                        {36,  4, 44, 12, 52, 20, 60, 28},
                        {35,  3, 43, 11, 51, 19, 59, 27},
                        {34,  2, 42, 10, 50, 18, 58, 26},
                        {33,  1, 41,  9, 49, 17, 57, 25}
                    };
```

‘’

3. 16位密码存储采用long long

```
long long KEY = 0x133457799BBCDFF1;
long long M   = 0x7CA66C6EEA3AF51F;
```

”

C语言源码重要函数

```
//将KEY 、M变为8*8
void num_to_set(long long key,int key_set[8][8]) {
    int a[64] = {0};
    int i = 0;
    for(i = 0;i < 64;i++) {
        a[64-1-i] = key % 2;
        key /= 2;
    }
    for(int m = 0;m < 8;m++) {
        for(int n = 0;n < 8;n++){
            key_set[m][n] = a[m*8+n];
        }
    }

}


//规则交换表置换操作
void set_to_plus(int set[8][8],int set_plus[8][8],int perm_key[8][8],int perm_num,int set_num) {
    int index = 0;
    for(int m = 0;m < 8;m++) {
        for(int n = 0;n < perm_num;n++){
            index = perm_key[m][n];
            set_plus[m][n] = set[(index-1)/set_num][(index-1)%set_num];
        }
    }
}
```

```
//将矩阵一分为二
void set_to_two(char mode) {
    int i = 0,j = 0;
    if(mode == 'c'){
        for(int m = 0;m < 4;m++) {
            for(int n = 0;n < 7;n++) {
                C[0][i] = key_set_plus[m][n];
                i++;
            }
        }
        for(int m = 4;m < 8;m++) {
            for(int n = 0;n < 7;n++) {
                D[0][j] = key_set_plus[m][n];
                j++;
            }
        }
    }else{
```

```
//将8*8矩阵变为long long
long long hh(int arr[8][8])
{
    long long result= 0;
    int i, j;
    long long tmp = 1;
    for (i = 7; i >=0; i--)
    {
        for (j = 7; j >= 0; j--)
        {
            result += tmp * arr[i][j];
            tmp *= 2;
        }
    }
    return result;
}


//左移
int* left_shift(int a[28],int num){
    int* b = (int *)malloc(28*sizeof(int));
    for(int i = 0;i < 28-num;i++){
        b[i] = a[i+num];
    }
    int j = 0;
    for(int i = 28-num;i < 28;i++){
        b[i] = a[j];
        j++;
    }
    return b;
}
```

```
//求子密钥
for(int i = 0;i < 16;i++){
    //左移相应位置
    int index = num_shift[i];
    int* cindex = left_shift(C[i],index);
    int* dindex = left_shift(D[i],index);
    for(int m = 0;m < 28;m++) {
        C[i+1][m] = cindex[m];
    }
    for(int m = 0;m < 28;m++) {
        D[i+1][m] = dindex[m];
    }

    //合并CnDn为一个8*7的矩阵
    CnDn_to_Kn(C[i+1],D[i+1],K[i+1]);

    //perm_K置换求子密钥
    index = 0;
    for(int m = 0;m < 8;m++) {
        for(int n = 0;n < 6;n++){
            index = perm_K[m][n];
            K_plus[i+1][m][n] = K[i+1][(index-1)/7][(index-1)%7];
        }
    }
    //set_to_plus(K[i+1],K_plus[i+1],index_set,6,8);

}
```

```
//求L1R1--L16R16
for(int x = 0;x < 16;x++){
    //R置换
    index = 0;
    for(int m = 0;m < 8;m++) {...
    }
    //Rn与异或Kn+1
    for(int m = 0;m < 8;m++) {...
    }
    //S盒置换
    for(int m = 0;m < 8;m++) {...
    }

    //P盒置换
    index = 0;
    for(int m = 0;m < 8;m++) {
        for(int n = 0;n < 4;n++){
            index = perm_P[m][n];
            R_plus[x][m][n] = R_plus_temp[x][(index-1)/4][(index-1)%4];
        }
    }

    //与Ln异或
    for(int m = 0;m < 8;m++) {...
    }

    //Rn+1=R_plus,Ln+1 = Rn
    for(int m = 0;m < 8;m++) {...
    }
    for(int m = 0;m < 8;m++) {...
    }
}
```

实验结果

加密过程

```
C:\Users\74156\Desktop>gcc des.c

C:\Users\74156\Desktop>a.exe
k1:000110 110000 001011 101111 111111 000111 000001 110010
k2:011110 011010 111011 011001 110110 111100 100111 100101
k3:010101 011111 110010 001010 010000 101100 111110 011001
k4:011100 101010 110111 010110 110110 110011 010100 011101
k5:011111 001110 110000 000111 111010 110101 001110 101000
k6:011000 111010 010100 111110 010100 000111 101100 101111
k7:111011 001000 010010 110111 111101 100001 100010 111100
k8:111101 111000 101000 111010 110000 010011 101111 111011
k9:111000 001101 101111 101011 111011 011110 011110 000001
k10:101100 011111 001101 000111 101110 100100 011001 001111
k11:001000 010101 111111 010011 110111 101101 001110 000110
k12:011101 010111 000111 110101 100101 000110 011111 101001
k13:100101 111100 010111 010001 111110 101011 101001 000001
k14:010111 110100 001110 110111 111100 101110 011100 111010
k15:101111 111001 000110 001101 001111 010011 111100 001010
k16:110010 110011 110110 001011 000011 100001 011111 110101
0BD82E1AB44A2346
```

解密过程

```
C:\Users\74156\Desktop>a.exe
k1:000110 110000 001011 101111 111111 000111 000001 110010
k2:011110 011010 111011 011001 110110 111100 100111 100101
k3:010101 011111 110010 001010 010000 101100 111110 011001
k4:011100 101010 110111 010110 110110 110011 010100 011101
k5:011111 001110 110000 000111 111010 110101 001110 101000
k6:011000 111010 010100 111110 010100 000111 101100 101111
k7:111011 001000 010010 110111 111101 100001 100010 111100
k8:111101 111000 101000 111010 110000 010011 101111 111011
k9:111000 001101 101111 101011 111011 011110 011110 000001
k10:101100 011111 001101 000111 101110 100100 011001 001111
k11:001000 010101 111111 010011 110111 101101 001110 000110
k12:011101 010111 000111 110101 100101 000110 011111 101001
k13:100101 111100 010111 010001 111110 101011 101001 000001
k14:010111 110100 001110 110111 111100 101110 011100 111010
k15:101111 111001 000110 001101 001111 010011 111100 001010
k16:110010 110011 110110 001011 000011 100001 011111 110101
0BD82E1AB44A2346
```