

5.2 对称加密算法逆向分析

@ 两种对称加密算法进行介绍

□ RC4

□ DES



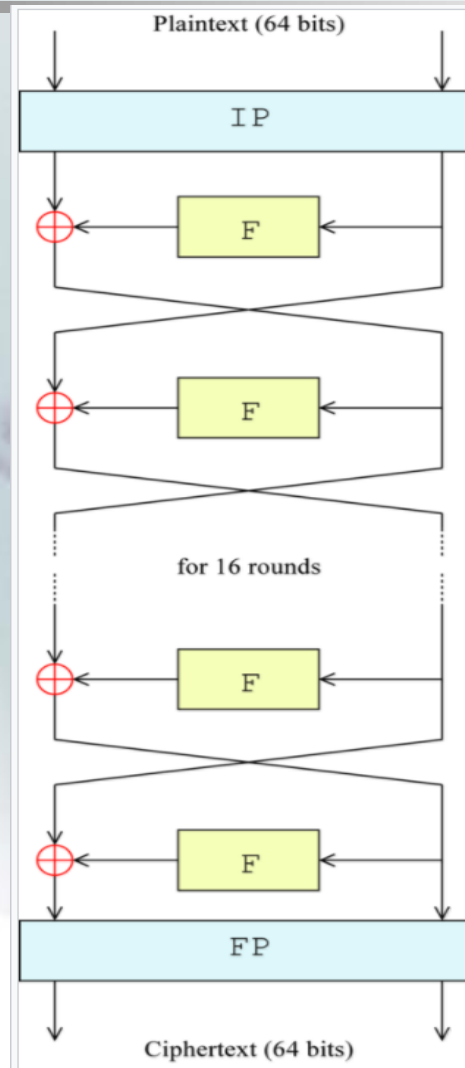
对称加密算法 DES

@ 原理介绍

- ❑ **DES**属于对称密码体制，加解密使用相同的密钥，有效密钥的长度为**56**位。
- ❑ **DES**为分组密码算法，分组长度为**64**位，使用**Feistel**的结构作为加解密的基本结构。



对称加密算法 DES



对称加密算法 DES

- ❑ 首先,将输入的**64**位明文进行一个初始置换（**IP**），并将得到的结果分为左右两个分组，各为**32**位。
- ❑ 进行初始置换后，对左右两个分组进行**16**轮相同轮函数的迭代，每轮迭代都有置换和代换。需要注意的是最后一轮迭代输出不进行左右两个分组的交换。
- ❑ 经**16**轮迭代后，得到的结果再经逆初始置换（**FP**）的作用后,作为加密的最终输出。



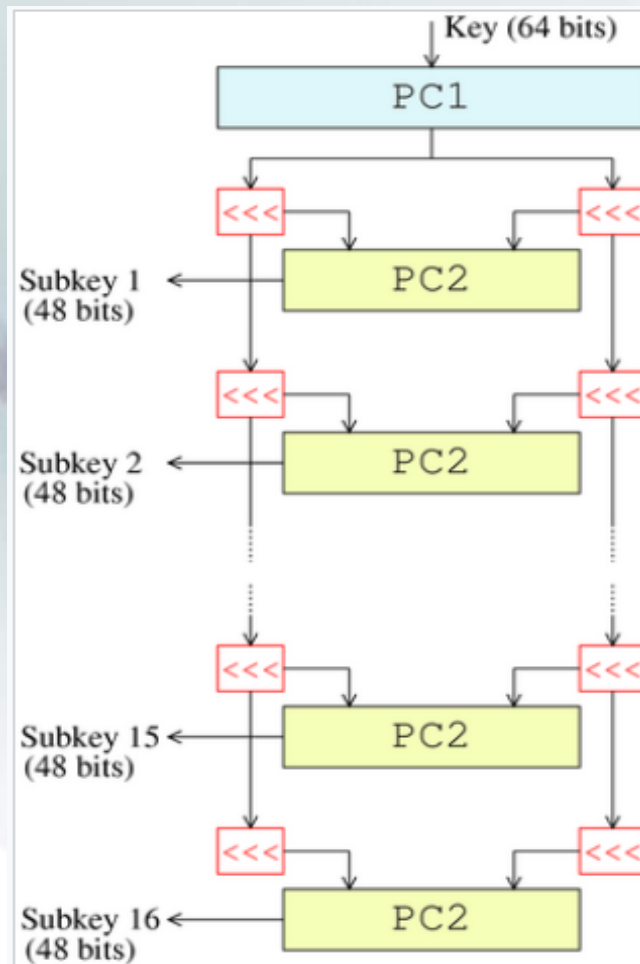
对称加密算法 DES

④ 密钥生成算法

- ❑ 在加密过程的每一轮迭代中，轮函数 F 需要在右边分组和每一轮的子密钥的控制下得到输出，并与左边分组进行异或操作。
- ❑ 每一轮子密钥就是在密钥生成算法的控制下产生的。



对称加密算法 DES



对称加密算法 DES

- ❑ 从图中可以看到，对于用户输入的**64**位密钥，先经过置换选择**PC-1**得到有效的**56**位密钥——剩下的**8**位要么直接丢弃，要么作为奇偶校验位。
- ❑ 然后将得到的**56**位密钥分为左右两个**28**位的半密钥。在接下来的**16**轮中，左右两个半密钥都被左移**1**或**2**位（具体由轮数决定），然后通过置换选择**PC-2**产生**48**位的子密钥。



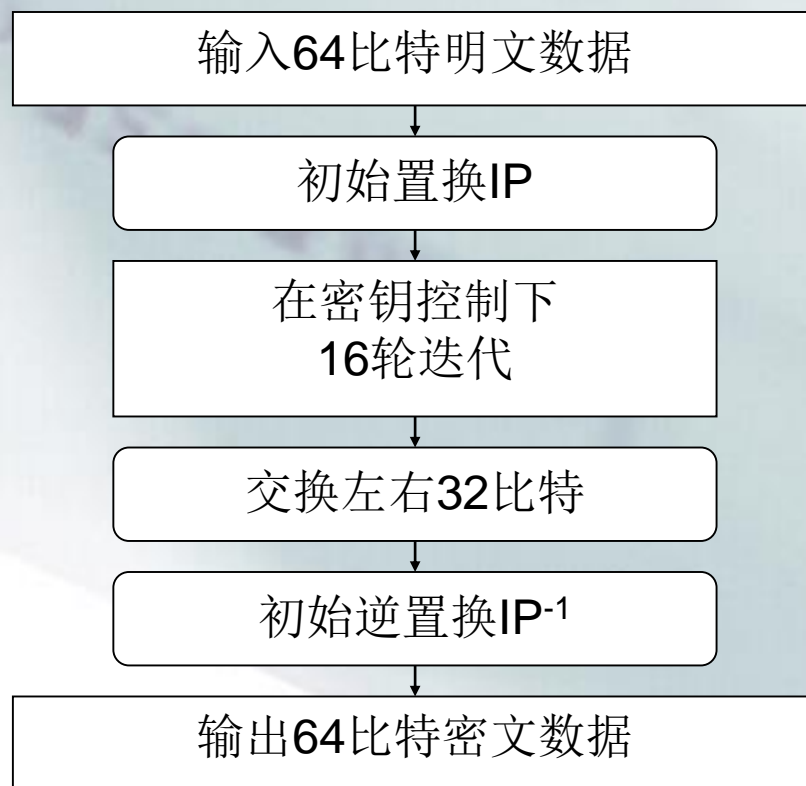
对称加密算法 DES

④ 加解密步骤



对称加密算法 DES

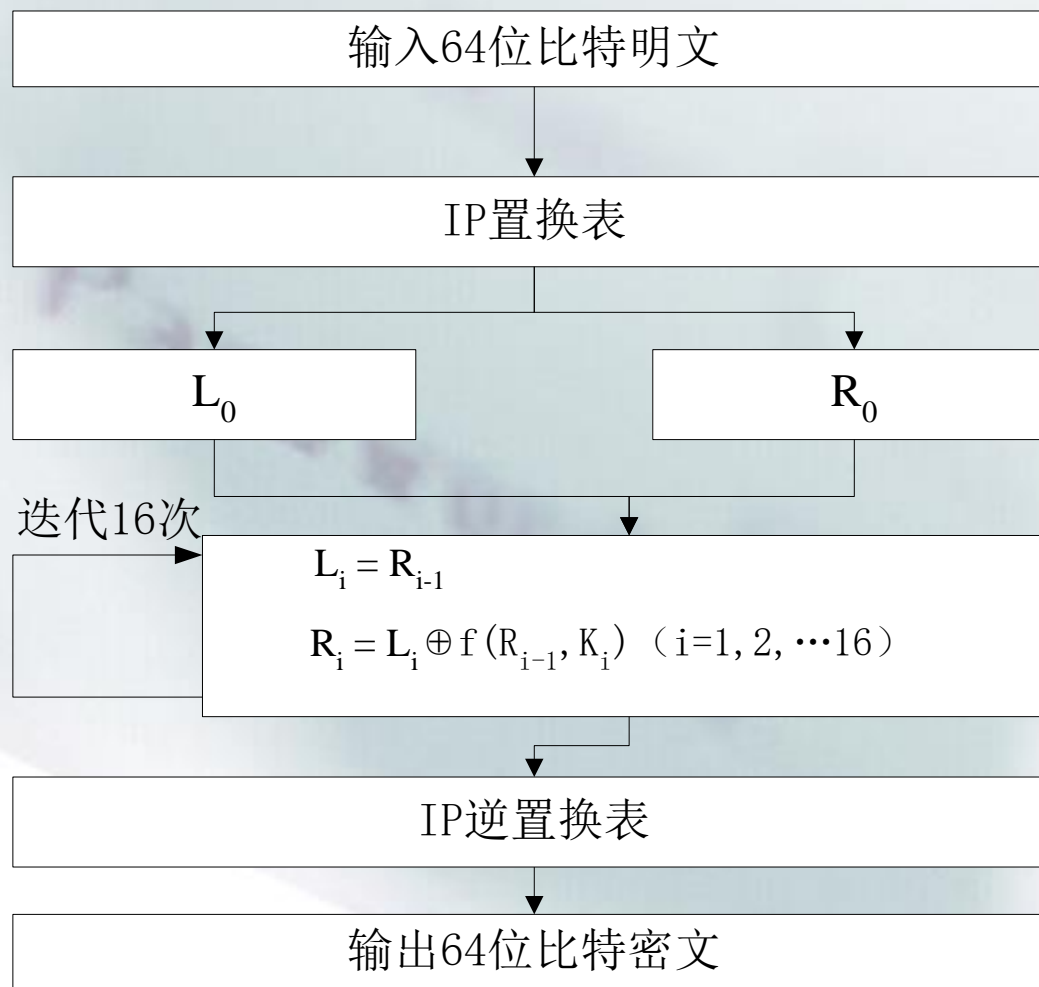
DES利用56比特串长度的密钥K来加密长度为64位的明文，得到长度为64位的密文



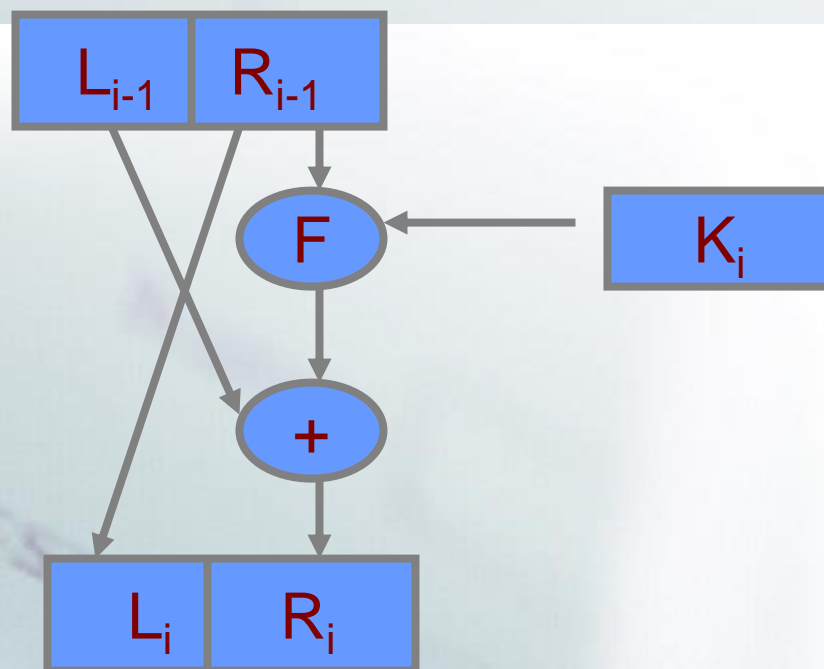
崔宝江



对称加密算法 DES



对称加密算法 DES



一轮加密的简图

对称加密算法 DES

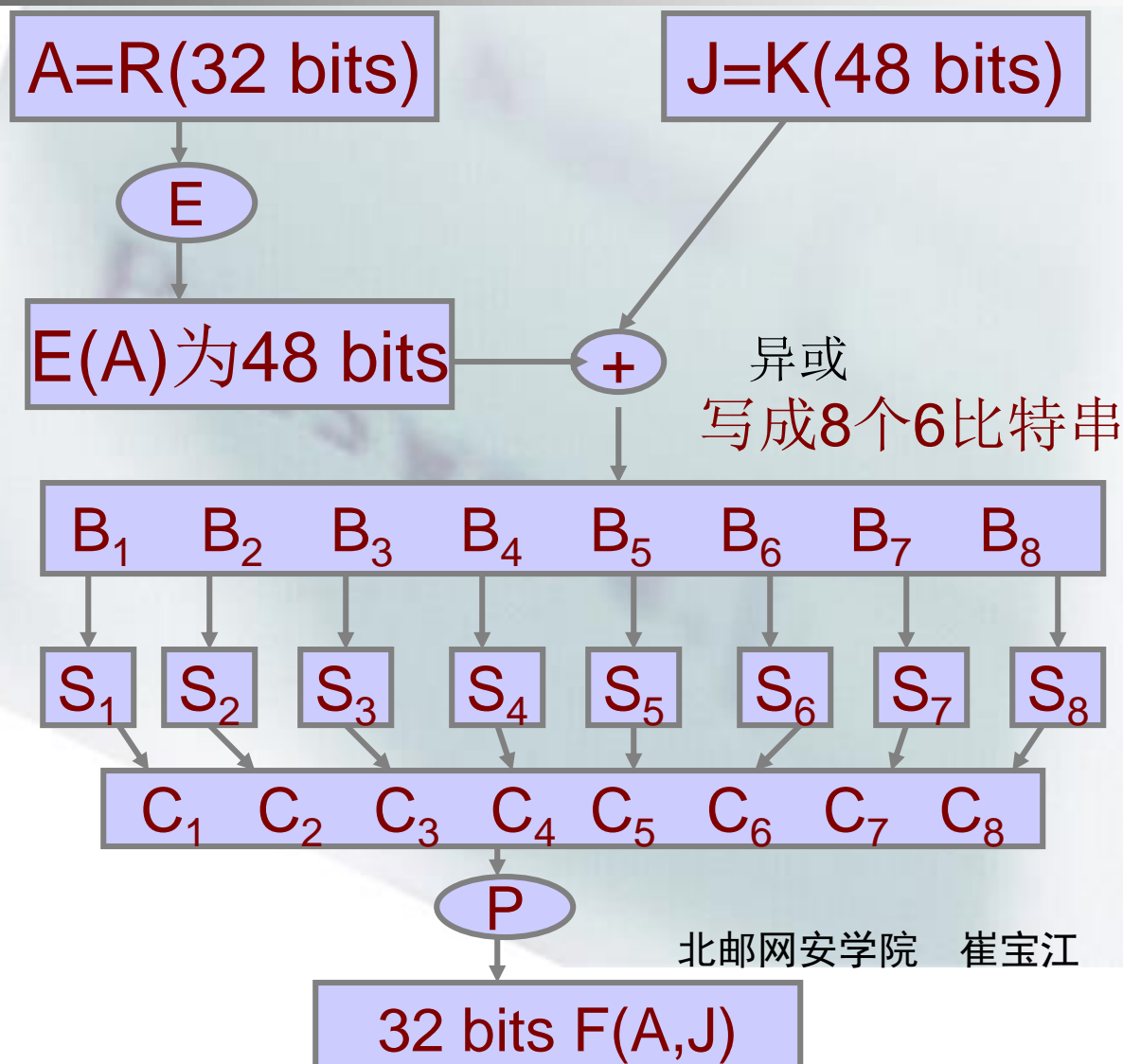
F函数说明

$F(R_{i-1}, K_i)$ 函数F以长度为32的比特串 $A=R$ (32bits) 作第一个输入, 以长度为48的比特串变元 $J=K$ (48bits) 作为第二个输入。产生的输出为长度为32的位串。

- (1) 对第一个变元A, 由给定的扩展函数E, 将其扩展成48位串 $E(A)$;
- (2) 计算 $E(A) + J$, 并把结果写成连续的8个6位串, $B=b_1b_2b_3b_4b_5b_6b_7b_8$;
- (3) 使用8个S盒, 每个 S_j 是一个固定的 4×16 矩阵, 它的元素取0~15的整数。给定长度为6个比特串, 如 $B_j=b_1b_2b_3b_4b_5b_6$, 计算 $S_j(B_j)$ 如下: b_1b_6 两个比特确定了 S_j 的行数, $r(0 \leq r \leq 3)$; 而 $b_2b_3b_4b_5$ 四个比特确定了 S_j 的列数 $c(0 \leq c \leq 15)$ 。最后 $S_j(B_j)$ 的值为S-盒矩阵 S_j 中 r 行 c 列的元素 (r, c) , 得 $C_j=S_j(B_j)$;
- (4) 最后, 进行固定置换P。



对称加密算法 DES



对称加密算法 DES

(1) 给定64位的密钥K，放弃奇偶校验位（8，16，...，64）并根据固定置换PC1来排列K中剩下的位。我们写

$$PC1(K)=C_0D_0$$

其中 C_0 由PC1(K)的前28位组成； D_0 由后28位组成；

(2) 对 $1 \leq i \leq 16$ ，计算

$$C_i = LS_i(C_{i-1})$$

$$D_i = LS_i(D_{i-1})$$

LS_i 表示循环左移2或1个位置，取决于 i 的值。
 $i=1,2,9$ 和 16 时移1个位置，否则移2位置；

(3) $K_i = PC2(C_iD_i)$ ，PC2为固定置换。北邮网安学院 崔宝江



对称加密算法 DES



对称加密算法 DES

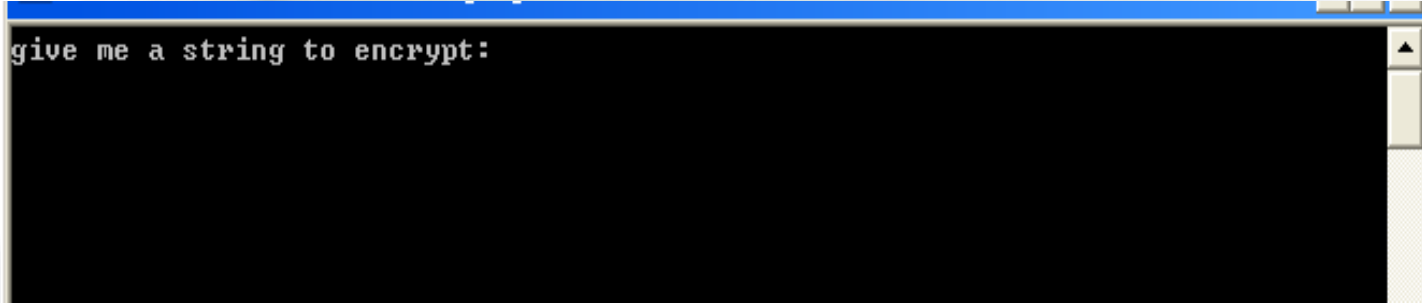
@ 逆向分析

- ❑ 通过分析一个**DES**加密的示例来加深一下对**DES**加密算法的理解
- ❑ 由于**DES**加密算法较为复杂，仍然在**F5**插件反编译得到的伪代码基础上进行分析，并且主要对一些关键的函数进行分析



对称加密算法 DES

- ❑ 运行 **desenc.exe**，发现程序需要我们输入一个字符串来加密



```
give me a string to encrypt:
```

对称加密算法 DES

- ❑ 使用IDA打开desenc.exe，定位到main函数，查看程序大致流程
 - 首先获取用户的输入，将其存放到分配的栈空间v9中；
 - 接着判断输入内容的长度是否符合要求；
 - 在长度符合要求的情况下将地址0x409070处的数据作为参数传入函数sub_401560中；
 - 接着将用户的输入作为参数传递给函数sub_4010B0；
 - 最后还有一个字节数组之间的比较



对称加密算法 DES

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    int v4; // eax@3
    char v5[8]; // [sp+4h] [bp-28h]@3
    int v6; // [sp+Ch] [bp-20h]@1
    int v7; // [sp+10h] [bp-1Ch]@1
    char v8; // [sp+14h] [bp-18h]@1
    char v9; // [sp+18h] [bp-14h]@1

    v6 = dword_409070;
    v7 = dword_409074;
    v8 = byte_409078;
    puts(aGiveMeAStringT);
    scanf(aS, &v9);
    if ( strlen(&v9) == 8 )
    {
        sub_401560(&v6);
        sub_4010B0(&v9, v5);
        v4 = 0;
        while ( v5[v4] == byte_409030[v4] )
        {
            if ( ++v4 >= 8 )
            {
                puts(aG00dJob);
                system(aPause);
                return 0;
            }
        }
    }
    return -1;
}
```



对称加密算法 DES

- 下面跟进函数 `sub_401560`，该函数的参数是一块固定的8字节数据，猜测是产生密钥的函数。该函数中又存在着多处函数调用

```
int __cdecl sub_401560(int a1)
{
    int v1; // ebx@1
    void *v2; // ebp@1
    int result; // eax@2
    char *v4; // edi@2
    char v5; // [sp+10h] [bp-A8h]@1
    char v6; // [sp+2Ch] [bp-8Ch]@2
    char v7; // [sp+48h] [bp-70h]@2
    char v8; // [sp+78h] [bp-40h]@1

    sub_4011E0(a1, (int)&v8, 8);
    sub_401490(&v8, &v5);
    v1 = 0;
    v2 = &unk_40B930;
    do
    {
        sub_4014F0(&v5, &v5, byte_408208[v1]);
        sub_4014F0(&v6, &v6, byte_408208[v1]);
        result = sub_4014C0(&v5, &v7);
        v4 = (char *)v2;
        v2 = (char *)v2 + 48;
        ++v1;
        qmemcpy(v4, &v7, 0x30u);
    }
    while ( (signed int)v2 < (signed int)&dwor_408C30 );
    return result;
}
```



对称加密算法 DES

- 对于函数调用**sub_4011E0**，将固定的数据作为参数传递给它，并将处理后的结果放入栈空间**v8**中，跟进该函数发现是该函数取出参数中每一字节数据的每一位，并将每一位数据再放入一个字节中；

```
1 int __cdecl sub_4011E0(int a1, int a2, int a3)
2 {
3     int result; // eax@1
4     int v4; // esi@1
5     int v5; // edi@2
6     signed int v6; // eax@3
7     signed int v7; // edx@3
8
9     result = a3;
10    v4 = 0;
11    if ( a3 > 0 )
12    {
13        v5 = a2;
14        do
15        {
16            v6 = 0;
17            v7 = 7;
18            do
19            {
20                *(_BYTE *)(v5 + v6++) = (*(_BYTE *)(v4 + a1) >> v7--) & 1;
21                while ( v6 < 8 );
22                result = a3;
23                ++v4;
24                v5 += 8;
25            } while ( v4 < a3 );
26        }
27        return result;
28    }
```



对称加密算法 DES

- 函数调用**sub_401490**则是将栈中的数据**v8**作为参数进行处理，一共进行了**56**轮循环，每一轮循环都是根据字节数组**0x4081A0**处的值来将**v8**中的数据放入输出地址**a2**处，经过该函数，输入的**64**字节数据变成了**56**字节数据，联想到**DES**生成密钥的过程我们知道这是**PC-1**置换选择

```
int __cdecl sub_401490(int a1, int a2)
{
    int result; // eax@1
    signed int v3; // esi@1
    int v4; // edx@2

    result = a2;
    v3 = 56;
    do
    {
        v4 = (&byte_4081A0[result++] - a2);
        --v3;
        *(_BYTE *)(result - 1) = *(_BYTE *)(v4 + a1 - 1);
    }
    while ( v3 );
    return result;
}
```



对称加密算法 DES

❑ 字节数组0x4081A0处的数据如下，与标准的PC-1表进行对照发现该处就是PC-1置换表

```
.rdata:004081A0 ; _BYTE byte_4081A0[56]
.rdata:004081A0 byte_4081A0      db 39h, 31h, 29h, 21h, 19h, 11h, 9, 1, 3Ah, 32h, 2Ah, 22h
.rdata:004081A0                                     ; DATA XREF: sub_401490+510
.rdata:004081A0      db 1Ah, 12h, 0Ah, 2, 3Bh, 33h, 2Bh, 23h, 1Bh, 13h, 0Bh
.rdata:004081A0      db 3, 3Ch, 34h, 2Ch, 24h, 3Fh, 37h, 2Fh, 27h, 1Fh, 17h
.rdata:004081A0      db 0Fh, 7, 3Eh, 36h, 2Eh, 26h, 1Eh, 16h, 0Eh, 6, 3Dh, 35h
.rdata:004081A0      db 2Dh, 25h, 1Dh, 15h, 0Dh, 5, 1Ch, 14h, 0Ch, 4
```



对称加密算法 DES

- 知道了函数sub_401560为生成子密钥的函数
 - 便能知道函数sub_4014F0就是循环移位的函数，分别对56位密钥中的左右两个子密钥进行移位。
 - 同样函数sub_4014C0应该就是PC-2置换选择函数了，跟进该函数进行验证，发现其中也有一个查表的操作，而表中的数据刚好是PC-2置换选择表

```
.rdata:004081D8 ; _BYTE byte_4081D8[48]
.rdata:004081D8 byte_4081D8      db 0Eh, 11h, 0Bh, 18h, 1, 5, 3, 1Ch, 0Fh, 6, 15h, 0Ah
.rdata:004081D8                                     ; DATA XREF: sub_4014C0+5↑o
.rdata:004081D8      db 17h, 13h, 0Ch, 4, 1Ah, 8, 10h, 7, 1Bh, 14h, 0Dh, 2
.rdata:004081D8      db 29h, 34h, 1Fh, 25h, 2Fh, 37h, 1Eh, 28h, 33h, 2Dh, 21h
.rdata:004081D8      db 30h, 2Ch, 31h, 27h, 38h, 22h, 35h, 2Eh, 2Ah, 32h, 24h
.rdata:004081D8      db 1Dh, 20h
```


对称加密算法 DES

@函数sub_4014F0就是循环移位的函数

```
1 int __cdecl sub_401560(int a1)
2 {
3     int v1; // ebx@1
4     void *v2; // ebp@1
5     int result; // eax@2
6     char *v4; // edi@2
7     char v5; // [sp+10h] [bp-A8h]@1
8     char v6; // [sp+2Ch] [bp-8Ch]@2
9     char v7; // [sp+48h] [bp-70h]@2
10    char v8; // [sp+78h] [bp-40h]@1
11
12    sub_4011E0(a1, (int)&v8, 8);
13    sub_401490(&v8, &v5);
14    v1 = 0;
15    v2 = &unk_40B930;
16    do
17    {
18        sub_4014F0(&v5, &v5, byte_408208[v1]);
19        sub_4014F0(&v6, &v6, byte_408208[v1]);
20        result = sub_4014C0(&v5, &v7);
21        v4 = (char *)v2;
22        v2 = (char *)v2 + 48;
23        ++v1;
24        qmemcpy(v4, &v7, 0x30u);
25    }
26    while ( (signed int)v2 < (signed int)&dword_40BC30 );
27    return result;
28 }
```



对称加密算法 DES

□ 下面返回main，看一下加密的函数sub_4010B0

```
int __cdecl main(int argc, const char **argv, const char **envp)
```

```
{
    int v4; // eax@3
    char v5[8]; // [sp+4h] [bp-28h]@3
    int v6; // [sp+Ch] [bp-20h]@1
    int v7; // [sp+10h] [bp-1Ch]@1
    char v8; // [sp+14h] [bp-18h]@1
    char v9; // [sp+18h] [bp-14h]@1
```

```
    v6 = dword_409070;
    v7 = dword_409074;
    v8 = byte_409078;
    puts(aGiveMeAStringT);
```

```
    scanf(aS, &v9);
```

```
    if ( strlen(&v9) == 8 )
```

```
    {
```

```
        sub_401560(&v6);
```

```
        sub_4010B0(&v9, v5);
```

```
        v4 = 0;
```

```
        while ( v5[v4] == byte_409030[v4] )
```

```
        {
```

```
            if ( ++v4 >= 8 )
```

```
            {
```

```
                puts(aG00dJob);
```

```
                system(aPause);
```

```
                return 0;
```

```
            }
```

```
        }
```

```
    }
```



对称加密算法 DES

- ❑ 下面跟进进行加密的函数**sub_4010B0**，函数的开始调用了函数**byte2Bits (sub_4011E0)**将用户的输入中的每一位（一共有位）提取出来转换成一个**64**字节的数组；
- ❑ 接下来调用函数**sub_401270**对这**64**字节数组进行处理，这就是加密过程中的**IP**置换；
- ❑ 接着调用了两次**memcpy**将置换后的数据分成两个**32**字节的数组，也即**16**轮迭代中的左右两个分组；



对称加密算法 DES

```
int __cdecl sub_4010B0(int a1, int a2)
{
    void *v2; // ebx@1
    char v4; // [sp+Ch] [bp-A0h]@1
    char v5; // [sp+2Ch] [bp-80h]@2
    char v6; // [sp+4Ch] [bp-60h]@1
    char v7; // [sp+6Ch] [bp-40h]@1
    char v8; // [sp+8Ch] [bp-20h]@1

    byte2Bits(a1, (int)&v7, 8);
    sub_401270(&v7, (int)&v7);
    qmemcpy(&v6, &v7, 0x20u);
    v2 = &unk_40B930;
    qmemcpy(&v4, &v8, 0x20u);
do
{
    sub_401430((int)&v4, &v5, (int)v2);
    sub_401400(&v5, (int)&v6, 32);
    v2 = (char *)v2 + 48;
    qmemcpy(&v6, &v4, 0x20u);
    qmemcpy(&v4, &v5, 0x20u);
}
while ( (signed int)v2 < (signed int)&unk_40BC00 );
sub_401430((int)&v4, &v5, (int)&unk_40BC00);
sub_401400(&v6, (int)&v5, 32);
qmemcpy(&v7, &v6, 0x20u);
qmemcpy(&v8, &v4, 0x20u);
sub_4012B0(&v7, &v7);
return sub_401230(&v7, a2, 8);
}
```



对称加密算法 DES

- ❑ 在将数据分成左右两个分组后应该就要进入**16**轮循环迭代了，观察到该函数有一个**do...while**循环，循环的起点是**0x40B930**，增加的步长为**48**，循环的终点为**0x40BC00**，经计算刚好是进行了**15**轮循环，而缺少的一轮加密则出现在了**do...while**结束之后的地方；
- ❑ **16**轮循环迭代，这也是**DES**加密的一个特征



对称加密算法 DES

```
int __cdecl sub_4010B0(int a1, int a2)
{
    void *v2; // ebx@1
    char v4; // [sp+Ch] [bp-A0h]@1
    char v5; // [sp+2Ch] [bp-80h]@2
    char v6; // [sp+4Ch] [bp-60h]@1
    char v7; // [sp+6Ch] [bp-40h]@1
    char v8; // [sp+8Ch] [bp-20h]@1

    byte2Bits(a1, (int)&v7, 8);
    sub_401270(&v7, (int)&v7);
    qmemcpy(&v6, &v7, 0x20u);
    v2 = &unk_40B930;
    qmemcpy(&v4, &v8, 0x20u);
    do
    {
        sub_401430((int)&v4, &v5, (int)v2);
        sub_401400(&v5, (int)&v6, 32);
        v2 = (char *)v2 + 48;
        qmemcpy(&v6, &v4, 0x20u);
        qmemcpy(&v4, &v5, 0x20u);
    }
    while ( (signed int)v2 < (signed int)&unk_40BC00 );
    sub_401430((int)&v4, &v5, (int)&unk_40BC00);
    sub_401400(&v6, (int)&v5, 32);
    qmemcpy(&v7, &v6, 0x20u);
    qmemcpy(&v8, &v4, 0x20u);
    sub_4012B0(&v7, &v7);
    return sub_401230(&v7, a2, 8);
}
```



对称加密算法 DES

- 下面分析最关键的函数 `sub_401430`，对于其中的函数 `sub_4012F0`，通过观察其参数，我们可以进行猜测，`a1` 为一个 32 字节的数据，`v5` 则是分配的栈上的空间，大小为 `0x30`（48）字节，因此该函数可能是一个扩展置换，可以跟进该函数进行验证

```
int __cdecl sub_401430(int a1, char *a2, int a3)
{
    int result; // eax@1
    char v4; // [sp+8h] [bp-50h]@1
    char v5; // [sp+28h] [bp-30h]@1

    sub_4012F0((const void *)a1, (int)&v5);
    sub_401400(&v5, a3, 48);
    sub_401330((int)&v5, (int)&v4);
    result = sub_4013C0(v4, (int)&v4);
    memcpy(a2, &v4, 0x20u);
    return result;
}
```



对称加密算法 DES

- ❑ 分析函数sub_401400，发现其主要做的是：
将第一个参数和第二个参数逐字节的相加，并将结果‘与’上0x1（也即取最低位），这其实就是异或的操作，也是标准**DES**加密过程中右边分组经扩展置换后与密钥进行异或的部分



对称加密算法 DES

- ❑ 根据**DES**加密的步骤不难知道，函数 **sub_401330** 是**S**盒代换的部分，主要是根据 **6bit** 数据查表得到一个 **4bit** 数据的过程。
- ❑ 对于函数 **sub_4013C0**，跟进去后发现仍然有一个查表的操作，也即**P**盒置换的过程。
- ❑ 后续的处理则是**DES**最后的逆**IP**置换，以及从字节中提取 **bit** 位的逆操作（这里为了查表方便将字节中的每一位都提取出来放入一个字节中，构成一个字节数组）。



对称加密算法 DES

- ❑ **DES**加密的主要过程就分析完了，在加密完成之后将得到的密文与**0x409030**处的字节数组（密文）进行比较。
- ❑ 我们现在知道了加密算法为**DES**，密钥为地址**0x409070**处的8字节数据“**DE3_En1C**”，在编写解密函数时，只需要将生成的**16**轮子密钥倒过来使用即。
- ❑ 因解密代码较长，仅列出关键部分



对称加密算法 DES

```
int main()
{
    unsignedchar key[9]="DE3_En1C";
    unsignedchar plaintext[20];
    unsignedchar
ciphertext[8]={0xef,0x34,0xd4,0xa3,0xc6,0x84,0xe4,0x23};
    get_subkey(key);
    decryption(ciphertext,plaintext);
    plaintext[8]='\0';
    printf("%s\n",plaintext);
    system("pause");
return0;
}
```



对称加密算法 DES

```
void decryption(unsignedchar* ciphertext,unsignedchar* plaintext){
int i;
unsignedchar array_ciphertext[64];
unsignedchar f_result[32];
unsignedchar left_array[32];
unsignedchar right_array[32];
    byte2Bit(ciphertext,array_ciphertext,8);
    ip_replace(array_ciphertext,array_ciphertext);
    memcpy(left_array,array_ciphertext,32);
    memcpy(right_array,array_ciphertext+32,32);
    for(i=15;i>0;--i){
        f_func(right_array,f_result,&subkey[i][0]);
        byteXOR(f_result,left_array,32);
        memcpy(left_array,right_array,32);
        memcpy(right_array,f_result,32);
    }
    f_func(right_array,f_result,&subkey[i][0]);
    byteXOR(left_array,f_result,32);
    memcpy(array_ciphertext,left_array,32);
    memcpy(array_ciphertext+32,right_array,32);
    fp_replace(array_ciphertext,array_ciphertext);
    bit2Byte(array_ciphertext,plaintext,8);
}
```



对称加密算法 DES

- ❑ 运行该解密函数，得到解密结果并对其进行验证

```
HarDd3s?  
请按任意键继续. . .
```

```
give me a string to encrypt:  
HarDd3s?  
G00d Job!!  
请按任意键继续. . .
```



Q & A

谢谢!

