

# 实验二 DES&AES

---

You can also access this document here: <http://xpty.xyz/post/desaes/>

课程名称: 《信息安全》SOFT130018.01

任课老师: 李景涛

助教: 杨鹏宇 24210240377@m.fudan.edu.cn  
张安琪 24212010048@m.fudan.edu.cn

## 实验目的

- Implementing DES algorithm
- Understanding usage of AES algorithm

## 实验内容

### How DES Works in Detail

DES is a **block cipher**--meaning it operates on plaintext blocks of a given size (64-bits) and returns ciphertext blocks of the same size. Thus DES results in a *permutation* among the  $2^{64}$  possible arrangements of 64 bits, each of which may be either 0 or 1. Each block of 64 bits is divided into two blocks of 32 bits each, a left half block **L** and a right half **R**.

DES operates on the 64-bit blocks using **key** sizes of 56 bits. The keys are actually stored as being 64 bits long, but every 8th bit in the key is not used and got eliminated when create **subkeys**.

The DES algorithm uses the following steps:

### Step 1: Create 16 subkeys, each of which is 48-bits long.

The 64-bit key is permuted according to the table, **PC-1**.

Next, split this key into left and right halves,  $C_0$  and  $D_0$ , where each half has 28 bits.

Create sixteen blocks  $C_n$  and  $D_n$ ,  $1 \leq n \leq 16$ . Each pair of blocks  $C_n$  and  $D_n$  is formed from the previous pair  $C_{n-1}$  and  $D_{n-1}$ , respectively, for  $n = 1, 2, \dots, 16$ , using **the schedule of "left shifts"** of the previous block. To do a left shift, move each bit one place to the left, except for the first bit, which is cycled to the end of the block.

Form the keys  $K_n$ , for  $1 \leq n \leq 16$ , by applying the permutation table **PC-2** to each of the concatenated pairs  $C_n D_n$ . Each pair has 56 bits, but **PC-2** only uses 48 of these.

## Step 2: Encode each 64-bit block of data.

There is an *initial permutation* **IP** of the 64 bits of the message data **M**.

Next divide the permuted block **IP** into a left half  $L_0$  of 32 bits, and a right half  $R_0$  of 32 bits.

Now proceed through 16 iterations, for  $1 \leq n \leq 16$ , using a function  $f$  which operates on two blocks--a data block of 32 bits and a key  $K_n$  of 48 bits--to produce a block of 32 bits.

Let  $+$  denote XOR addition, (bit-by-bit addition modulo 2). Then for  $n$  going from 1 to 16 we calculate:

$$\begin{aligned} L_n &= R_{n-1} \\ R_n &= L_{n-1} + f(R_{n-1}, K_n) \end{aligned}$$

This results in a final block, for  $n = 16$ , of  $L_{16}R_{16}$

We then **reverse** the order of the two blocks into the 64-bit block  $R_{16}L_{16}$  and apply a final permutation  $IP^{-1}$

Finally output the result in hexadecimal format.

Decryption is simply the inverse of encryption, following the same steps as above, but reversing the order in which the subkeys are applied.

## How the function $f$ works

To calculate  $f$ , we first expand each block  $R_{n-1}$  from 32 bits to 48 bits. This is done by using a selection table that repeats some of the bits in  $R_{n-1}$ . We'll call the use of this selection table the function **E**. Thus  $E(R_{n-1})$  has a 32 bit input block, and a 48 bit output block.

Next in the  $f$  calculation, we XOR the output  $E(R_{n-1})$  with the key  $K_n$ :

$$K_n + E(R_{n-1})$$

We now have 48 bits, or eight groups of six bits. For each group of six bits: we use them as addresses in tables called "**S boxes**". Each group of six bits will give us an address in a different **S** box. Located at that address will be a 4 bit number. This 4 bit number will replace the original 6 bits. The net result is that the eight groups of 6 bits are transformed into eight groups of 4 bits (the 4-bit outputs from the S boxes) for 32 bits total.

Write the previous result, which is 48 bits, in the form:

$$K_n + E(R_{n-1}) = B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8$$

where each  $B_i$  is a group of six bits.

Now calculate:

$$S_1(B_1) S_2(B_2) S_3(B_3) S_4(B_4) S_5(B_5) S_6(B_6) S_7(B_7) S_8(B_8)$$

where  $S_i(B_i)$  refers to the output of the  $i$ -th **S** box.

The first and last bits of  $B$  represent in base 2 a number in the decimal range 0 to 3 (or binary 00 to 11). Let that number be  $i$ . The middle 4 bits of  $B$  represent in base 2 a number in the decimal range 0 to 15 (binary 0000 to 1111). Let that number be  $j$ . Look up in the table the number in the  $i$ -th row and  $j$ -th column. It is a number in the range 0 to 15 and is uniquely represented by a 4 bit block. That block is the output  $S(B)$  of  $S$  for the input  $B$ .

The final stage in the calculation of  $f$  is to do a permutation **P** of the **S**-box output to obtain the final value of  $f$ :

$$f = P(S_1(B_1) S_2(B_2) \dots S_8(B_8))$$

## How to use AES in OpenSSL or Python

**OpenSSL** (recommended) is a robust, commercial-grade, and full-featured toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. It is also a *general-purpose cryptography library*.

You can use the command line tool **Enc** for encryption and decryption. **Enc** is used for various block and stream ciphers using keys based on passwords or explicitly provided:

- <https://wiki.openssl.org/index.php/Enc>

**PyCryptodome** is a self-contained Python package of low-level cryptographic primitives.

You can also use the **AES** module it provided:

- <https://www.pycryptodome.org/en/latest/src/cipher/aes.html>

## 实验步骤与过程

实现如下四个DES函数：

1.

```
def permutation_by_table(block, block_len, table):  
    '''block of length block_len permuted by table  
    table'''
```

2.

```
def generate_round_keys(C0, D0):
    '''returns dict of 16 keys (one for each round)'''
    lrot_values = (1,1,2,2,2,2,2,2,2,1,2,2,2,2,2,1)
    # initial rotation

    # create 16 more different key pairs
    # WRITE YOUR CODE HERE!

    #form the keys from concatenated CiDi 1<=i<=16 and by
    applying PC2
    # WRITE YOUR CODE HERE!
    return round_keys
```

3.

```
def round_function(Ri, Ki):
    # expand Ri from 32 to 48 bit using table E
    Ri = permutation_by_table(Ri, 32, E)

    # xor with round key
    # WRITE YOUR CODE HERE!

    # split Ri into 8 groups of 6 bit
    # WRITE YOUR CODE HERE!

    # interpret each block as address for the S-boxes
    # WRITE YOUR CODE HERE!

    # pack the blocks together again by concatenating
    # WRITE YOUR CODE HERE!

    # another permutation 32bit -> 32bit
    Ri = permutation_by_table(Ri, 32, P)
```

4.

```
def encrypt(msg, key, decrypt=False):
    # permute by table PC1
    key = permutation_by_table(key, 64, PC1) # 64bit ->
    PC1 -> 56bit

    # split up key in two halves
    # WRITE YOUR CODE HERE!

    # generate the 16 round keys
    round_keys = generate_round_keys(C0, D0) # 56bit ->
    PC2 -> 48bit

    msg_block = permutation_by_table(msg, 64, IP)
    # WRITE YOUR CODE HERE!
```

```

# apply round function 16 times (Feistel Network):
# WRITE YOUR CODE HERE!

# final permutation
cipher_block = permutation_by_table(cipher_block, 64,
IP_INV)

return cipher_block

```

## 5. 解密DES密文

学号尾号为奇数的同学，解密 `cipher_text_odd` ；  
 学号尾号为偶数的同学，解密 `cipher_text_even` ；  
 key 都是 `FudanNiu`

```

cipher_text_odd = 0x9b99d07d9980305e
#cipher_text_even = 0x6f612748df99a70c

decrypt(cipher_text_odd, key)
#decrypt(cipher_text_even, key)

```

## 6. 解密AES密文

用第5步DES解密的结果作为AES的密钥，来解密AES密文。

学号尾号为奇数的同学，解密文件 `odd.enc` ；  
 学号尾号为偶数的同学，解密文件 `even.enc` ；

```

openssl enc -d -aes256 -in odd.enc -out odd.plain
openssl enc -d -aes256 -in even.enc -out even.plain

```

## 实验要求和评分

- 编程语言、编译运行时、所用工具等实验环境原则上不限，建议在给出的Python程序框架基础上补完。
  - 如使用其他实验环境，需要完成同等任务，并在实验报告中写明。
- 评分内容如下：（100分是本实验的总分，本学期五个实验各100分，学期末会加权得出实验分总分并汇入总成绩）

内容	总分100
permutation_by_table	5
generate_round_keys	25
round_function	25
encrypt	15
DES decryption	10
AES decryption	10
Document(实验报告)	10

实现各种加密模式(mode), 3DES, 可有适当加分.

## 实验提交

- 独立完成 project , 不分组
- 提交截止时间 : 4月13日 23:59 前
- 提交内容清单 :
  - 实验报告 (按照**实验报告模板**书写, 要求提交pdf格式文件) (命名格式: 姓名+学号+实验二.pdf)
  - 项目源代码 (命名格式: 学号\_des.py)
- 提交方式:  
所有提交清单中的文件,压缩后, eLearning提交
- 鼓励录制短视频, 介绍代码结构、演示运行结果、分析计算量等

## 参考资料

- [The DES Algorithm Illustrated](#) by J. Orlin Grabbe
- <https://www.openssl.org/>
- <https://www.pycryptodome.org/>