**DES算法实现**

**实验报告**

姓名：宋晓彤

学号：16340192

方向：嵌入式软件与系统

2018.10.18

# 原理概述

DES算法是一种对称加密算法，明文和密钥均由64位二进制码构成，其中，密钥每个字节的最后一位为奇偶校验位，保证一个字节中1的个数为奇数，在实际加密过程中没有起到作用。

DES算法通过置换、移位等操作，将原始密钥转化为16个子密钥，进而与置换后的明文进行异或、S盒选择、P盒置换等操作，再次交换、置换后产生与原始明文完全不同的输出，从而完成加密过程。解密过程与加密过程近似，不同在于应调换16个子密钥的顺序，所以只要接收方得到正确的密钥，便可对得到的加密后二进制串进行解码得到传输的原始明文。

# 总体结构与流程

DES算法具体流程如下：

1. IP置换原始明文并分解：使用IP置换表将明文各二进制码的顺序进行调换，置换过后将得到的IP二进制码分解成两部分，左右两部分各32位；
2. 置换原始密钥并分解：使用PC\_1置换规则将原始密钥置换成新密钥，此时，密钥的奇偶校验位已被省去，然后再将得到的56位密钥分解成两部分，前后各28位；
3. 生成16位子密钥：将第二步得到的两部分按照左移表分别进行连续的移位，每次移位后将两部分合并，经过PC\_2置换得到一个子密钥，最终得到16个子密钥，每个子密钥为48位；
4. 循环操作：首先将明文的右半部分作为新的左半部分，再经过E置换扩展成48位，与第一个子密钥异或后，进入S盒进行选择，输出32位二进制码，再把其进入P盒后置换得到的新32位二进制码与旧的左半部分异或，得到结果作为新的右半部分，循环16次操作得到最后的左右两部分；
5. 交换并置换：将最后得到的第16个明文的循环结果的左右部分交换，进行IP的逆置换过后，得到的结果即为加密输出。

# 模块分解

根据DES算法的流程，我们将算法的过程分为如下几个功能模块：置换并分解，子密钥生成，feistel轮函数，交换并置换。

## 【模块一 置换并分解】

由于明文和密钥在一开始都要经过置换和分解的过程，所以此模块输入为明文/密钥、置换规则，输出为分解得到的左半部分和右半部分。其中，明文的置换规则为IP，密钥的置换规则为PC\_1。

在代码中用transforming()函数表示置换过程，用trans\_half()函数表示置换及分解的全过程。

## 【模块二 子密钥生成】

子密钥生成的部分可以划分为两部分，第一步是经过移位将密钥的分解结果分别移位并组合得到16个56位的二进制码，每个二进制码再通过PC\_2置换的缩位得到48位的子密钥。

在代码中用shift\_16()函数表示移位得到16个56位二进制的过程，用key\_16()函数表示生成子密钥的过程。

## 【模块三 轮函数】

轮函数由E置换、S盒选择、P盒置换三个部分构成，输入起始明文拆分后的左右两部分，得到最后第16个输出的新的左右两部分

在代码中用recycle()函数表示轮函数。

## 【模块四 交换并置换】

将模块三的输出的左右两部分交换后进行逆置换，得到结果

在代码中使用相同的置换方法得到输出。

# 数据设计与文件结构

## 【文件结构】

运行文件------------------------------------------------main.py

函数文件---------------------------------------------function.py

置换及盒规则文件-------------------------------------rule.py

## 【数据设计】

置换规则均由列表存储，列表中元素为数字类型，表示放入其所在位置的被置换数据的第n个元素，置换规则分别为IP \_IP PC\_1 PC\_2 E P。

16位左移位数也由列表存储，列表中元素为数字类型，表示每一次移位的多少，规则为left\_shift。

s盒用列表存储，列表中元素为以数字位数据类型的列表，作为每6位输入得到4位输出结果的十进制表示。

以上内容均在rule.py文件中。

简单函数分别为shift()左移函数，exclusive()异或函数，transforming()置换函数，较为复杂的函数有s\_box()选择函数，shift\_16()产生16个成对的移位输出，key\_16()产生16个子密钥输出，recycle()轮函数。

以上内容均在function.py文件中。

在main.py文件中，data为原始明文，key为原始密钥；lr是一个大小为2的列表，分别表示明文分解或每次轮循环时产生的左右两部分；c0，d0表示原始密钥第一次分解产生的左右两部分；c，d为两个大小为17的列表，分别表示c0，d0和其他16对移位输出；k为大小为16的列表，表示16个子密钥；res为最终产生的结果。

# 代码实现

## 【function.py】

# 置换并分解

# data----要处理的数据

# trans---置换规则

# len-----新得到的二进制位数

*def* tran\_half(*data*, *trans*, *len*):

res = [['' for i in range(0, len//2)] for i in range(0, 2)]

for i in range(0, 2):

for j in range(0, len//2):

res[i][j] = data[trans[i\*(len//2)+j]-1]

return res[0], res[1]

# c0 d0-----第一次分解密钥的两部分

# left------左移规则

# bit-------左右两部分的大小（32）

# len-------循环次数

*def* shift\_16(*c0*, *d0*, *left*, *bit*, *len*):

c = [['' for i in range(bit)] for i in range(len+1)]

d = [['' for i in range(bit)] for i in range(len+1)]

c[0] = c0;

d[0] = d0;

for i in range(1, len+1):

c[i] = shift(c[i-1], left[i-1])

d[i] = shift(d[i-1], left[i-1])

return c, d

# 产生16个子密钥

# c d-------左右两部分的列表

# num-------循环次数

# trans-----置换规则

# len-------子密钥位数

*def* key\_16(*c*, *d*, *num*, *trans*, *len*):

k = [['' for i in range(len)] for i in range(num)]

for i in range(num):

k[i] = transforming(c[i+1]+d[i+1], trans, len)

return k

# s盒选择

# data-------处理数据

# s----------s盒的列表，元素仍为列表

# inputlen, output\_len-------输入6位，输出4位

# times------操作次数8

*def* s\_box(*data*, *s*, *input\_len*, *output\_len*, *times*):

res = ['' for i in range(output\_len \* times)]

for i in range(times):

x = *int*(data[i\*input\_len])\*2 + *int*(data[i\*input\_len+5])

y = *int*(data[i\*input\_len+1])\*8 + *int*(data[i\*input\_len+2])\*4 + *int*(data[i\*input\_len+3])\*2 + *int*(data[i\*input\_len+4])

binary = bin(s[i][x\*16 + y])

for j in range(4):

if len(binary)-1-j >= 0:

if binary[len(binary)-1-j] != 'b':

res[(i+1)\*output\_len-1-j] = binary[len(binary)-1-j]

else:

res[(i+1)\*output\_len-1-j] = '0'

else:

res[(i+1)\*output\_len-1-j] = '0'

return res

# 轮函数

# l r-------一轮循环被操作的左右两部分

# k---------当前要用的单个子密钥

# ex\_len----扩展位数48

# s---------整个s盒

# len-------输出位数

*def* recycle(*l*, *r*, *k*, *ex\_len*, *s*, *len*):

res = [['' for i in range(len)] for i in range(2)]

res[0] = r

ex\_r\_k = exclusive(transforming(r, rule.E\_Trans, ex\_len), k, ex\_len)

s\_32 = s\_box(ex\_r\_k, s, 6, 4, 8)

p\_32 = transforming(s\_32, rule.P, 32)

res[1] = exclusive(l, p\_32, 32)

return res

## 【main.py】

数据

# set l0 and r0

lr = function.tran\_half(data, rule.ip, 64)

# set c0 and d0

c0, d0 = function.tran\_half(key, rule.pc\_1, 56)

# left shift the cds

c = [['' for i in range(28)] for i in range(17)]

d = [['' for i in range(28)] for i in range(17)]

c, d = function.shift\_16(c0, d0, rule.left\_shift, 28, 16)

# 16 key[0-15]

k = function.key\_16(c, d, 16, rule.PC\_2, 48)

加密

#16 operations

for r in range(16):

lr = function.recycle(lr[0], lr[1], k[r], 48, rule.S, 32)

#change l and r then do the transform

res = function.transforming(lr[1]+lr[0], rule.\_ip, 64)

解密

lr = function.tran\_half(res, rule.ip, 64)

for r in range(16):

lr = function.recycle(lr[0], lr[1], k[15-r], 48, rule.S, 32)

res = function.transforming(lr[1]+lr[0], rule.\_ip, 64)

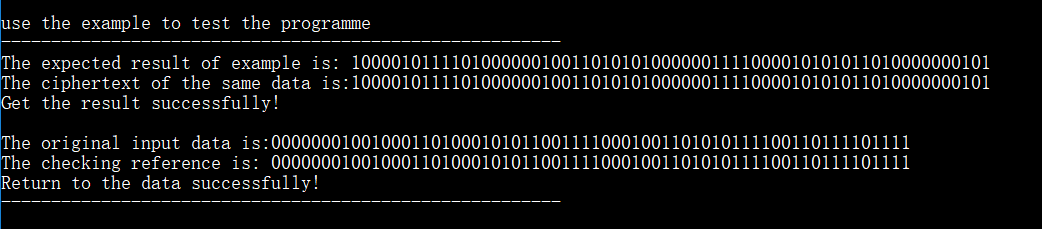
# 实验结果

在网络上查询我们可以得到一对样例，当明文是00000001001000110100010101

10011110001001101010111100110111101111，密钥是0001001100110100010101110

111100110011011101111001101111111110001时，输出为100001011110100000010

0110101010000001111000010101011010000000101，我们使用程序输出加密后的cipher和解密结果的checking reference和输出结果与原始明文进行对比，我们可以发现实验成功。



自输入数据，得到解码结果判断是否成功解码，仍旧得到试验成功的结果。