# 一、DES

题目链接：

<https://buuoj.cn/match/matches/15/challenges#Yusa%E7%9A%84%E5%AF%86%E7%A0%81%E5%AD%A6%E8%AF%BE%E5%A0%82%E2%80%94%E2%80%94SEDSED>

题目来源平台：BUUCTF

题目名称：Yusa的密码学课堂——SEDSED

题目描述：签完到就开始认真上课了（提交'flag{}'所包裹字符串）

题目类型：CRYPTO

题目附件：



FLAG：

# 二、解题工具

|  |  |  |  |
| --- | --- | --- | --- |
| 工具类型 | 工具名称 | 版本 | 备注 |
| 基础环境 | Python | 3.7 |  |
|  |  |  |

# 三、解题过程

### 步骤1

先分析代码：

1. 是一道DES模式的密码题
2. 由此段代码可以分析出，这里的DES只进行了两轮

|  |
| --- |
| def encrypt(plain\_text, sub\_keys):  plain\_textb = ""  for i in range(16):  plain\_textb += '{0:04b}'.format(int(plain\_text[i],16))  plain\_textp = permutation(plain\_textb)  left,right = plain\_textp[:32],plain\_textp[32:]  out = func(right,sub\_keys[0])  temp = int(out,2) ^ int(left,2)  left,right = right,'{0:032b}'.format(temp)  out = func(right,sub\_keys[1])  temp = int(out,2) ^ int(left,2)  left = '{0:032b}'.format(temp)  final = inv\_permutation(left+right)  cipher = hex(int(final,2))[2:]  return cipher |

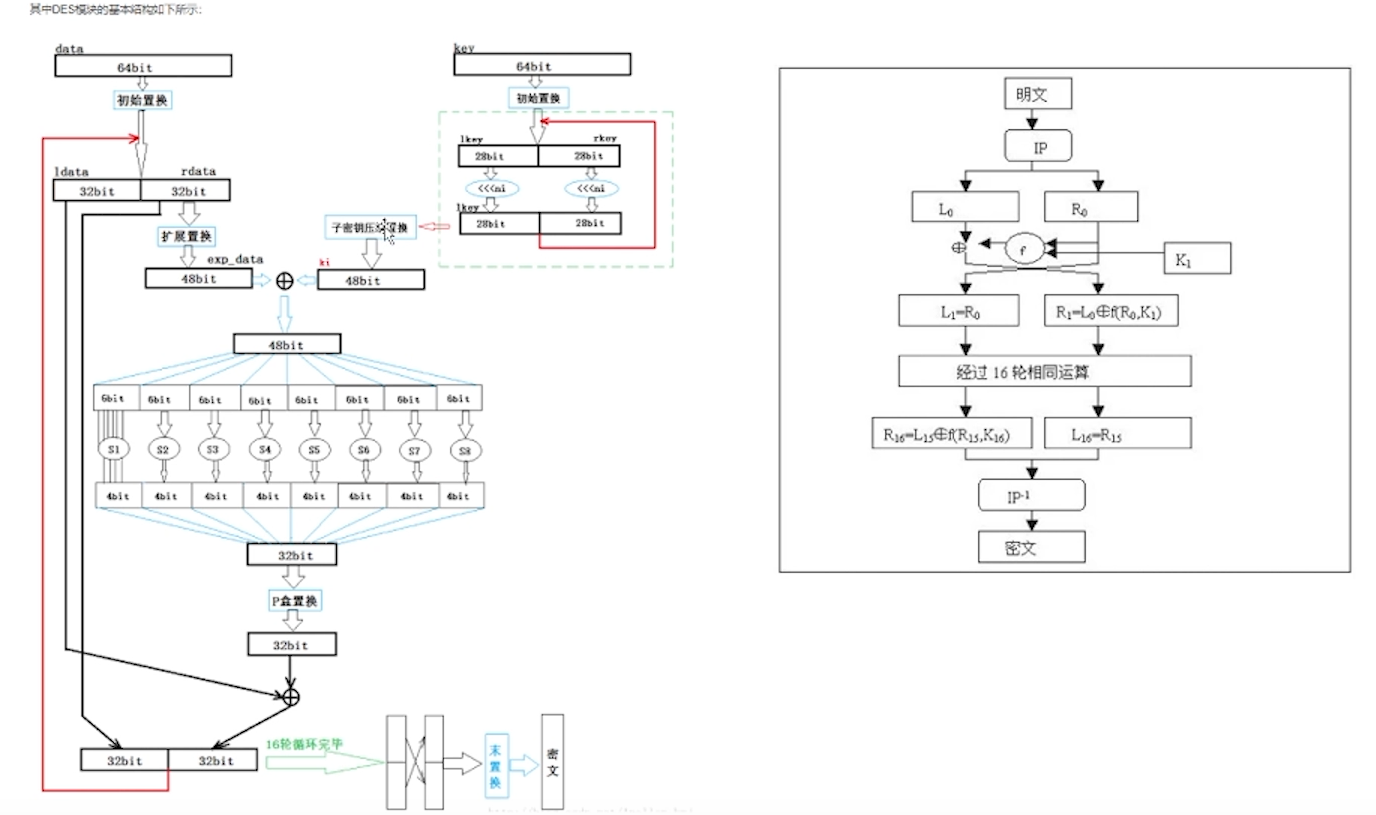
1. 由此段代码可以看出：

此处是对“testtest”进行一个加密，输出，然后再对flag进行加密。所以已知一对明文密文和flag的密文。

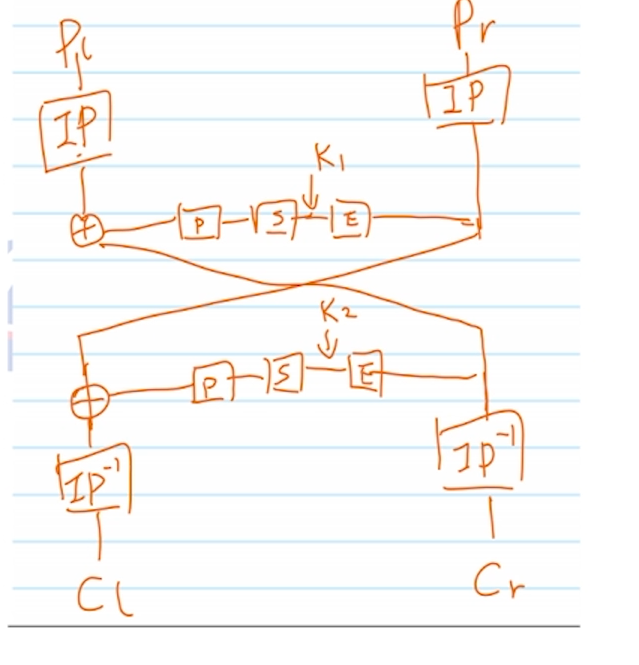
|  |
| --- |
| taststring = "testtest"  print(enc2(taststring.encode().hex(),key\_64))  with open("flag.txt") as f:  flag = f.read()  print("flag: ",myenc(flag.encode().hex())) |

### 步骤2

理解DES模式：



所以可以将本题简化成：



我们需要做的就是对密钥的解密。即将密文Cr进行IP置换和明文Pl的IP置换进行异或，然后通过P盒的逆，再通过S盒，因为有八个S盒，所以获得八组的四个输出，即4\*\*8个可能。然后将明文Pr进行IP置换，再进行拓展置换得到一个输出。接下来是对密钥的验证，因为密钥是从56比特压缩成48比特，所以需要进行一个2\*\*8的爆破。再将得到的密钥对testtest进行一个加密。如果匹配则就可以对flag进行一个解密。

### 步骤3

代码

明文密文的置换：

|  |
| --- |
| cipher=bin(int('77a35598c47aeea6',16))[2:].rjust(64,'0')  c\_iv=c.permutation(cipher)  c\_iv=bytes.fromhex(hex(int(c\_iv,2))[2:].rjust(16,'0'))  L=c\_iv[:4]  R=c\_iv[4:]  P=b'testtest'  P=bin(int(P.hex(),16))[2:].rjust(64,'0')  P\_p=c.permutation(P)  P\_l=bytes.fromhex(hex(int(P\_p[:32],2))[2:])  P\_r=P\_p[32:] |

异或

|  |
| --- |
| t1= bin(int(xor(P\_l,R).encode('latin1').hex(),16))[2:].rjust(32,'0') |

P盒的逆

|  |
| --- |
| t1\_ivp=unpermutation(t1) |

Key的获取

|  |
| --- |
| for i in range(8):  res=[]  s\_out=t1\_ivp[4\*i:4\*i+4]  s\_out=int(s\_out,2)  for j in range(4):  tmp=""  if s\_out in S[i][j]:  hang=j  lie=S[i][j].index(s\_out)  hang=bin(hang)[2:].rjust(2,'0')  lie=bin(lie)[2:].rjust(4,'0')  tmp=hang[0]+lie+hang[1]  res.append(tmp)  resualt.append(res) |

Key的恢复

|  |
| --- |
| P\_e=c.expand(P\_r)  KEY=[]  for i in range(8):  ikey=[]  m=P\_e[6\*i:6\*i+6]  for eres in resualt[i]:  ekey=int(m,2)\*int(eres,2)  ikey.append(bin(ekey)[2:].rjust(6,'0'))  KEY.append(ikey)  print(KEY) |

验证

|  |
| --- |
| def brute\_force(keyfinal,PP,CC):  temp=PP  cipher=CC  for x in range(2\*\*8):  digits=bin(x)[2:].rjust(8,'0')  digits=list(digits)  counter=0  key=list(keyfinal)  for i in range(56):  if key[i] =='x':  key[i]=digits[counter]  counter=counter+1  key\_56=''.join(key)  key\_64=['0']\*64  i=0  for u in KEY\_P1:  key\_64[u-1]=key[i]  i+=1  key\_64=''.join(key\_64)  key\_64=hex(int(key\_64,2))[2:].rjust(16,'0')  if c.enc2(temp,key\_64) == cipher:  print("Find euql key!")  print(key\_64)  flag = '86721c7c1ebe2d0af8aa8e073073931b4a5ae6dcf03c784e3c70b5f8ce71cf9eb87f9b836eea0118'  print(c.dec(flag,key\_64))  exit()  return |