密码学课程设计_验收回答模板

姓名：苏展　　学号：18271329

1.2 假设你手里有一段 Caesar 密文，请思考如何进行破解，还原出明文，描述思路即可。

将密文放入程序，然后写一个循环，偏移量从小到大，分别输出，然后查看哪一段可能的解密文比较像真正的明文.

2.3　RC4 是非线性加密，非线性体现在哪些部件或步骤?

非线性主要体现在 S 盒. 我们利用密钥 K 打乱了 S 盒，使之非线性

3.1 请截图展示实现 DES 轮变换的函数代码。

这里展示了轮变换函数 F_Func

以及其中调用的 Transform 函数、Xor 函数、S 盒变换函数

截图 1：

```
def F_Func(self,In,Ki):
    MR = self.Transform(In,self.E_Table,48)
    MR = self.Xor(MR,Ki)
    In = self.S_Func(In,MR)
    In = self.Transform(In,self.P_Table,32)
    return In

def S_Func(self,Out,In):
    Out=[]
    for i in range(8):
        j=(In[0+6*i]<<1)+In[5+6*i]
        k=(In[1+6*i]<<3)+(In[2+6*i]<<2)+(In[3+6*i]<<1)+In[4+6*i]
        Sbox=list('{:0>4s}'.format(bin(self.S_Box[i][j][k])[2:]))
        for i in range(len(Sbox)):
            Sbox[i]=int(Sbox[i])
        Out+=Sbox
    return Out
```

截图 2：

```
def Transform(self, In, Table, len):
    Temp=[]
    for i in range(len):
        Temp.append(In[Table[i]-1])
    return Temp
```

截图 3：

```
def Xor(self,InA,InB):
    for i in range(len(InA)):
        InA[i]=InB[i] ^ InA[i]
    return InA
```

3.3 请截图展示实现 DES 加密与解密有差异部分的代码。

加解密由 flag 控制

主要的区别在于加密时轮次由 0 到 15

解密时轮次由 15 到 0

```
if Flag == 1:
    M = self.Transform(M,self.IP_Table,64)
    for i in range(16):
        Temp = Ri
        Ri=self.F_Func(Ri,self.SubKey[i])
        Ri = self.Xor(Ri,Li)
        Li=Temp
    Li,Ri = Ri,Li
    M = Li + Ri
    M = self.Transform(M,self.IPInv_Table,64)
else:
    M = self.Transform(M,self.IP_Table,64)
    for i in range(15,-1,-1):
        Temp = Ri
        Ri=self.F_Func(Ri,self.SubKey[i])
        Ri = self.Xor(Ri,Li)
        Li=Temp
    Li,Ri = Ri,Li
    M = Li + Ri
    M = self.Transform(M,self.IPInv_Table,64)
```

4.1 请截图展示 SM4 中实现密钥扩展的函数代码，并解释数组 K 的作用

截图 1：

```
#生成子秘钥
for i in range(16):
    self.SK[i // 4] += self.key[i] << (24 - 8 * (i % 4))
self.SetRoundKey()
```

截图 2：

```
def SetRoundKey(self):
    for i in range(4):
        self.K[i] = self.SK[i] ^ self.FK[i]
    for i in range(32):
        self.K[i + 4] = self.K[i] ^ self.T1(self.K[i + 1] ^ self.K[i + 2] ^ self.K[i + 3] ^ self.CK[i])
        self.RK[i] = self.K[i + 4]
```

数组 K 的作用：

前四位保存最初的由 SK 和 FK 异或产生的四位 K 值

由于 RKi=Ki+4

所以 K 是中间密钥，用于递推生成轮密钥

5.1 请截图展示实现大整数模幂运算的函数代码。

```
def fast_power(base, power, n):
    """
    快速模幂运算
    """
    result = 1
    tmp = base
    while power > 0:
        if power&1 == 1:
            result = (result * tmp) % n
        tmp = (tmp * tmp) % n
        power = power>>1
    return result
```

5.4 请分别截图展示实现 RSA 加密和 RSA 解密的最关键的一句程序代码。

对于加密来说应该是这句

```
self.c.append(str(self.fast_power(ord(ms) , self.nume, self.numN)))
```

变量及方法解释 ：

Ms: 这句加密之前我使用了循环,变量 ms 中存放了每一个明文的字符. 并且用 ord 方法将其转换为 asc 码

Self.nume : 公钥 e,

numN : 随机生成的大素数 pq 的积

Self.fast_power : 自建方法, 可以快速进行模幂运算

Self.c : 密文

对于解密来说是这一句

```
self.m1.append(chr(self.fast_power(cs , self.numd, self.numN)))
```

变量及方法解释 :

Cs : 这句解密之前我同样使用了循环, 变量 cs 中存放了每一个密文的编码

Self.numd : 私钥

Chr : 模幂运算后的结果为 asc 码, 使用 chr 函数将其转换为字符

Self.m1: 解密文