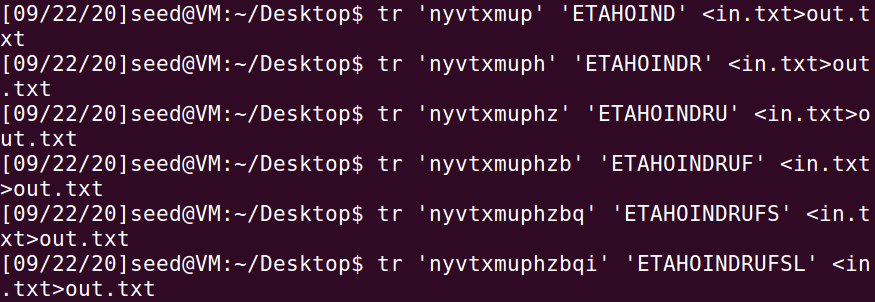
**网络空间安全实训Lab[7]**

**实验报告**

**57118117 谌雨阳**

**Task1:Frequency Analysis Against Monoalphabetic Substitution Cipher**

1、根据wiki上的频率分布推测出前几个常用字母及双字母、三字母组合，再结合猜词法进行后续推断，部分推断过程如下：

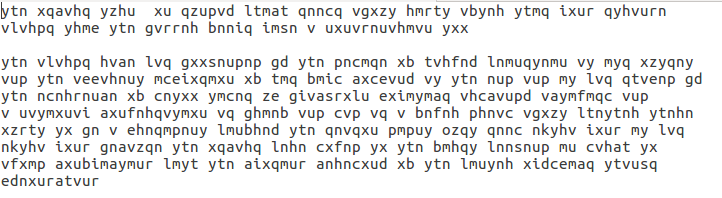


2、最终推断出完整的替换单表：

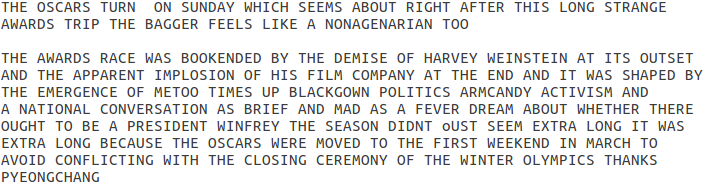


3、附部分密文与明文的对比：

密文：



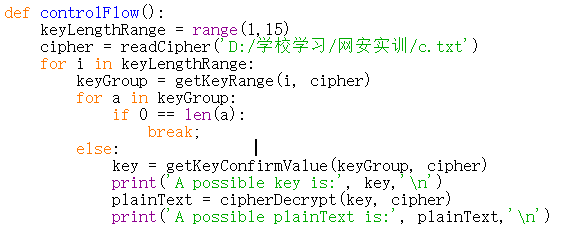
明文：



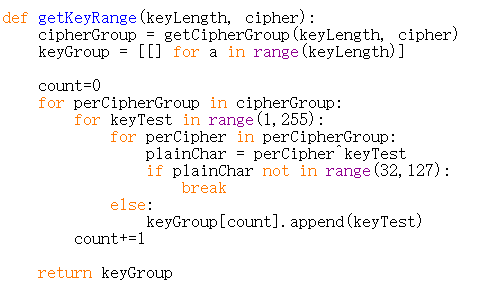
**破解维吉尼亚密码：**

1、题目的关键点在于，明文一定是由ASCII码可见字符段组成的，并且密钥一定存在。如果在假定的长度下，某一组密文在这个范围内找不到合理的密钥，即落在不可见字符段，那就说明密钥不可能是这个长度。

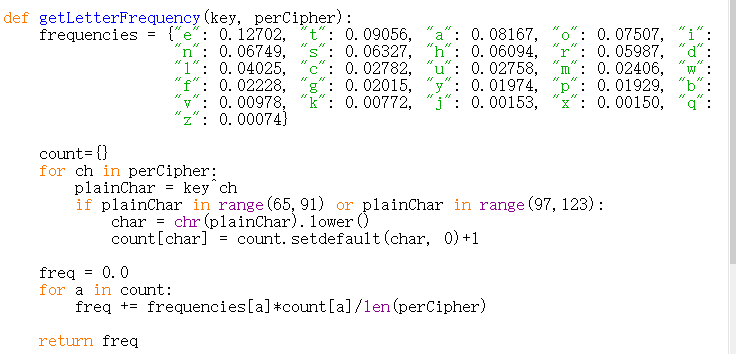
通过以下代码确认密钥长度：



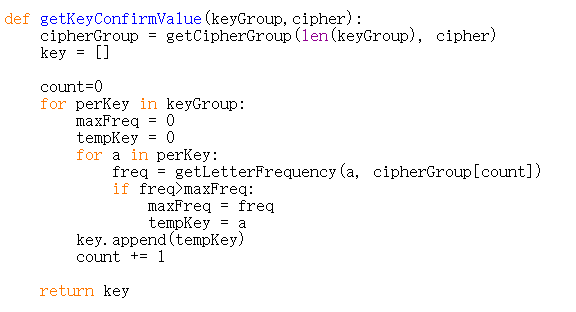
2、接下来对每一位密钥求其可能取值：



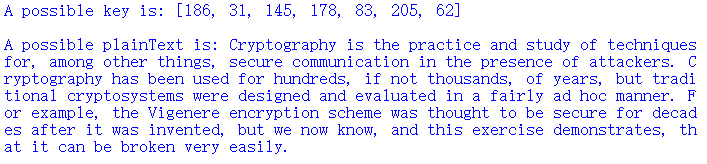
3、写入综合字频：



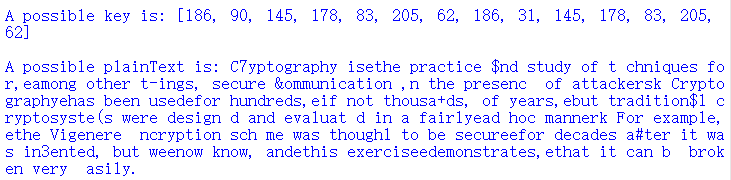
4、求解密出的明文的综合字频进行比较：



5、运行程序，解密成功：



该程序还解密出一种错误版本，也放在这里：



附python源代码：

def readCipher(filename):

file = open(filename, 'r')

strCipher = file.read()

cipher = []

index = 0

while index < len(strCipher):

cipher.append(int(strCipher[index:index+2], 16))

index += 2

return cipher

def getCipherGroup(keyLength, cipher):

cipherGroup = [[] for a in range(keyLength)]

count = 0

while count < len(cipher):

cipherGroup[(count) % keyLength] += [cipher[count]]

count += 1

return cipherGroup

def getKeyRange(keyLength, cipher):

cipherGroup = getCipherGroup(keyLength, cipher)

keyGroup = [[] for a in range(keyLength)]

count=0

for perCipherGroup in cipherGroup:

for keyTest in range(1,255):

for perCipher in perCipherGroup:

plainChar = perCipher^keyTest

if plainChar not in range(32,127):

break

else:

keyGroup[count].append(keyTest)

count+=1

return keyGroup

def getLetterFrequency(key, perCipher):

frequencies = {"e": 0.12702, "t": 0.09056, "a": 0.08167, "o": 0.07507, "i": 0.06966,

"n": 0.06749, "s": 0.06327, "h": 0.06094, "r": 0.05987, "d": 0.04253,

"l": 0.04025, "c": 0.02782, "u": 0.02758, "m": 0.02406, "w": 0.02360,

"f": 0.02228, "g": 0.02015, "y": 0.01974, "p": 0.01929, "b": 0.01492,

"v": 0.00978, "k": 0.00772, "j": 0.00153, "x": 0.00150, "q": 0.00095,

"z": 0.00074}

count={}

for ch in perCipher:

plainChar = key^ch

if plainChar in range(65,91) or plainChar in range(97,123):

char = chr(plainChar).lower()

count[char] = count.setdefault(char, 0)+1

freq = 0.0

for a in count:

freq += frequencies[a]\*count[a]/len(perCipher)

return freq

def getKeyConfirmValue(keyGroup,cipher):

cipherGroup = getCipherGroup(len(keyGroup), cipher)

key = []

count=0

for perKey in keyGroup:

maxFreq = 0

tempKey = 0

for a in perKey:

freq = getLetterFrequency(a, cipherGroup[count])

if freq>maxFreq:

maxFreq = freq

tempKey = a

key.append(tempKey)

count += 1

return key

def cipherDecrypt(key, cipher):

plainText = ''

index = 0

for a in cipher:

plainText += chr(key[index%len(key)]^a)

index += 1

return plainText

def controlFlow():

keyLengthRange = range(1,15)

cipher = readCipher('D:/学校学习/网安实训/c.txt')

for i in keyLengthRange:

keyGroup = getKeyRange(i, cipher)

for a in keyGroup:

if 0 == len(a):

break;

else:

key = getKeyConfirmValue(keyGroup, cipher)

print('A possible key is:', key,'\n')

plainText = cipherDecrypt(key, cipher)

print('A possible plainText is:', plainText,'\n')

if \_\_name\_\_=='\_\_main\_\_':

controlFlow()

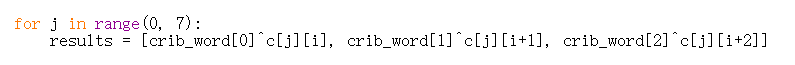
**一次一密密码：**

在课程中我们了解到，在英文句子的三元组中，the是最频繁出现的三元组。以此为线索采用如下方式破解一次一密密码。

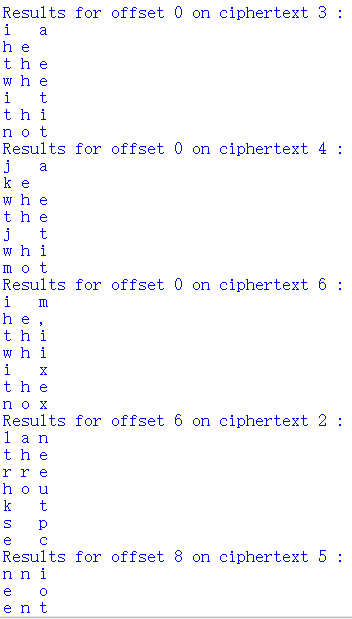
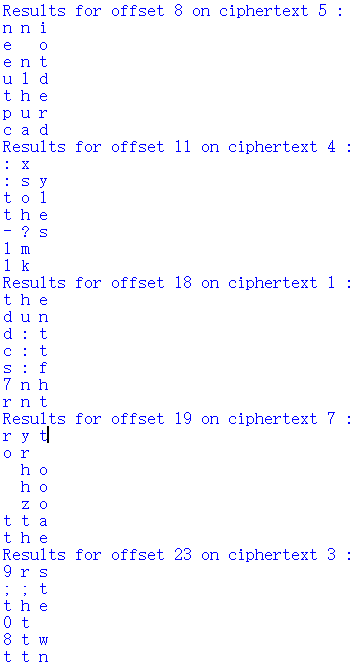
1、用the组成的ASCII码数组



2、和七组密文进行异或运算，解密寻找句子中可能能够解密成the的部分和位置：

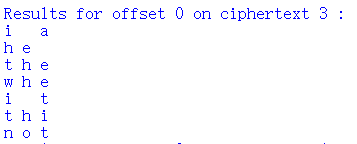


3、能够得出如下多组结果：

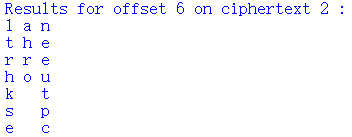
 

4、从中寻找比较符合句子结构和部分英文单词排列的几组：

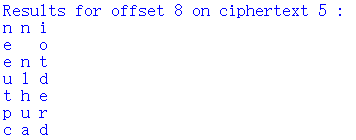
0：



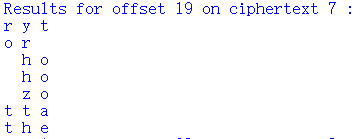
6：



8：



19：



60位16进制ASCII码对应30个字符，列出如下已知线索（推测）：

0 1 2 3 4 5 6 7 8 9 10 11 ... 19 20 21 ...

c1 i \_ a l a n n i r y t

c2 h e t h e \_ o o r \_

c3 t h e r r e n t \_ h o

c4 w h e h o u l d \_ h o

c5 i \_ t k \_ t h e \_ z o

c6 t h i s \_ p u r t t a

c7 n o t e \_ c a d t h e

接下来进入单词推断阶段：

由7、8、9、10、11课退出c4/6为s，得出一列；可再由c3推断出5为c，结合前后可推出c5/4为h，则前11位推断完成。

此后依次根据only,than,should,person,follow,than和mission可推出完整句子。

最终结果为：

I\_am\_planning a secret mission.

He is the only person to trust.

The current plan is top secret.

When should we meet to do this?

I think they should follow him.

This is purer than that one is.

Not one cadet is better than 1.