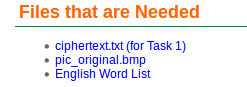
Lab7-report

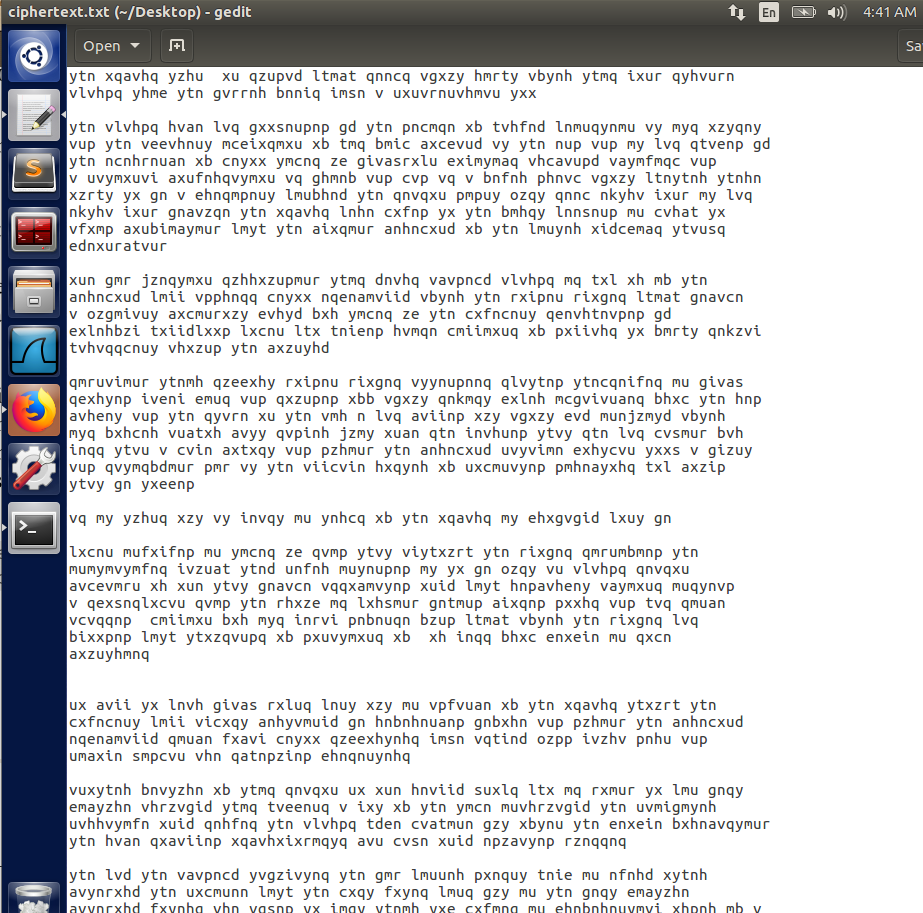
57118113 蔡义涵

**2.1 Task 1: Frequency Analysis Against Monoalphabetic Substitution Cipher**

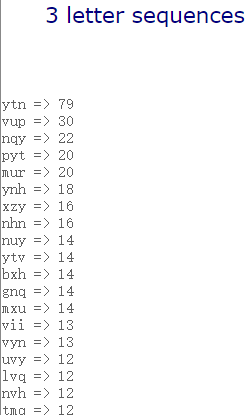
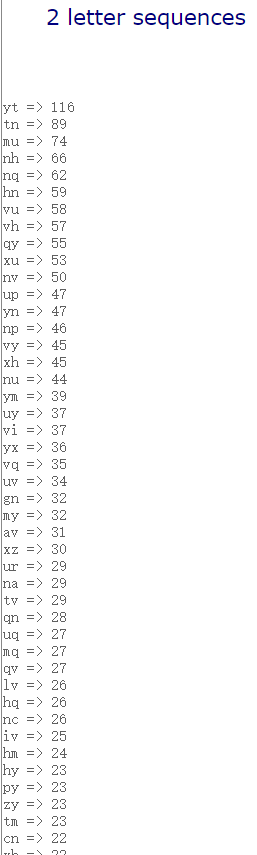
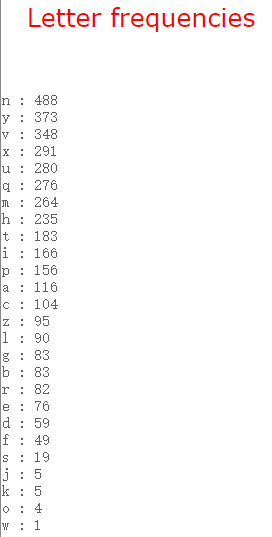
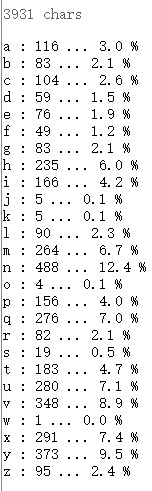
首先，我们进入seed实验室网站下载ciphertext.txt



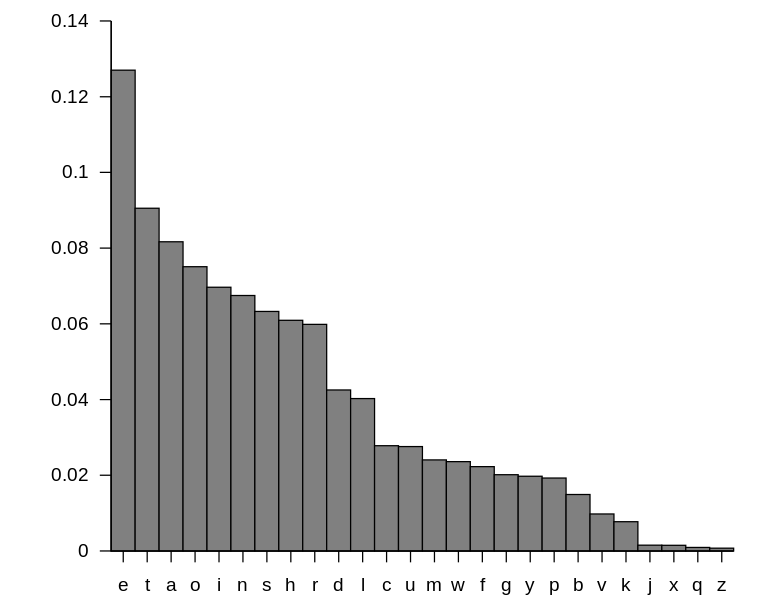
下载得到文件为

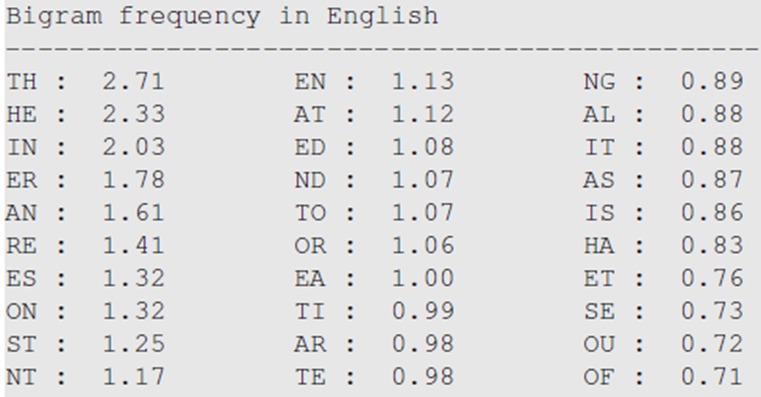


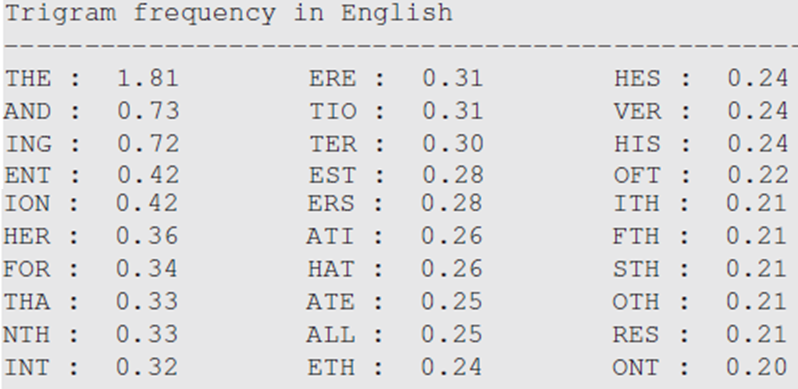
将ciphertext.txt文本复制到网站<http://www.richkni.co.uk/php/crypta/freq.php>中，进行频率分析：



由于维基百科访问限制，我们在国内网站和课堂ppt中找到了英语单字母、双字母和三字母使用频率表







根据单字母频率分析，大胆猜测加密方案为 ETAO nyvx

根据双字母频率分析，大胆猜测TH yt，HE tn，IN mu，ER nh

根据三字母频率分析，大胆猜测THE ytn，AND vup

目前猜测有

A v

D p

E n

H t

I m

N u

O x

R h

T y

再根据文本双字母频率表中nq排名第五，猜测ES nq，即S q

以及文本三字母频率表中mur排名第五，猜测ING mur，即G r

目前猜测有

A v

D p

E n

G r

H t

I m

N u

O x

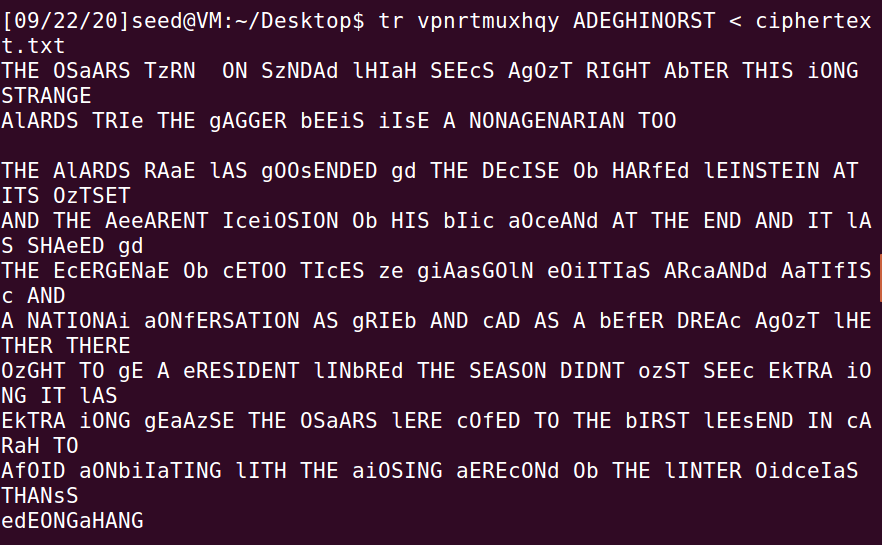
R h

S q

T y

在命令行中输入以下命令：

Tr vpnrtmuxhqy ADEGHINORST < ciphertext.txt

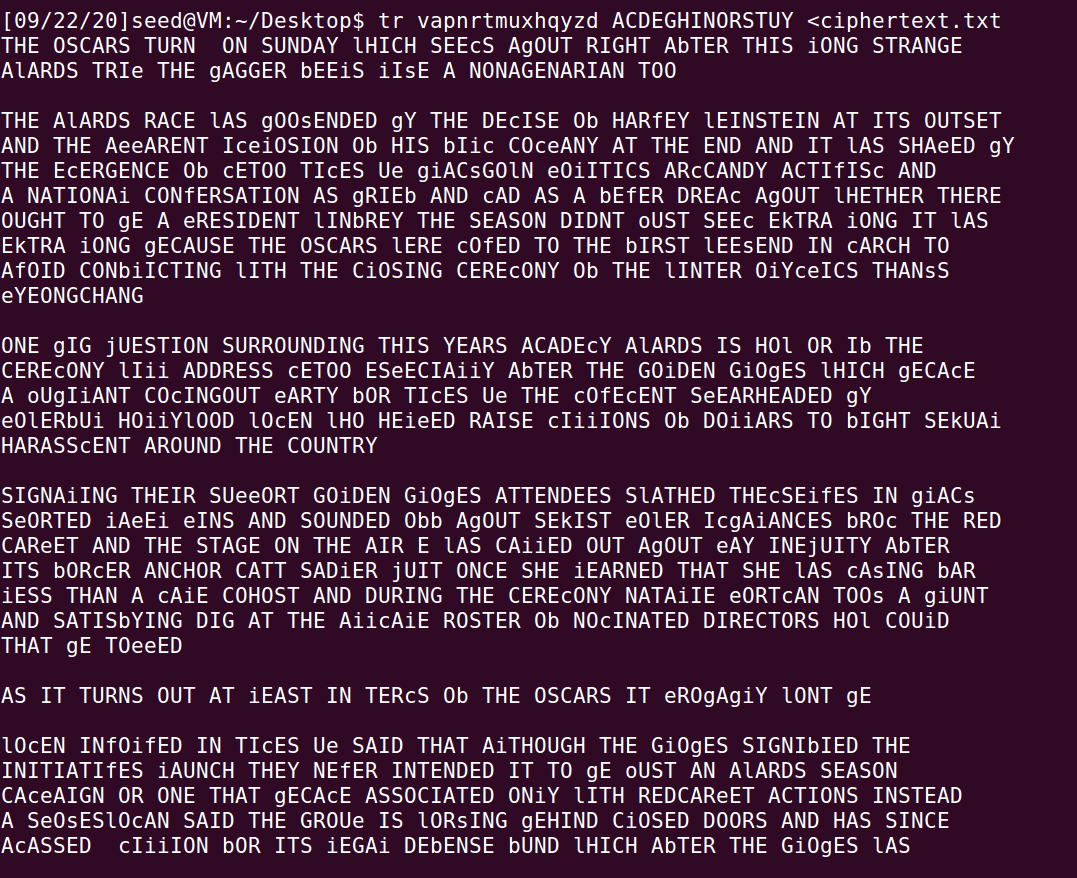


根据结果，猜测第三个单词为turn，猜测U z

猜测第二个单词为oscars，猜测C a

猜测第五个单词为sunday，猜测U z，Y d

重复命令，得到



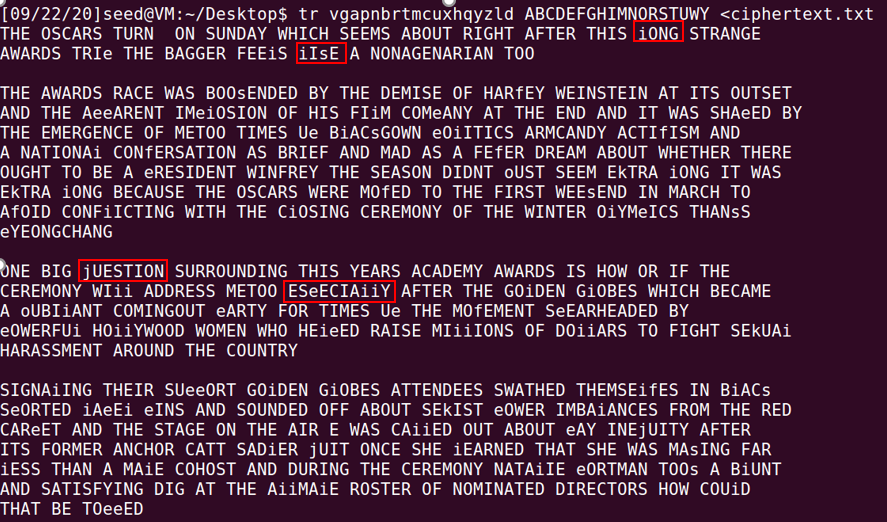
猜测第六个单词为which，猜测W l

猜测第七个单词为seems，猜测M c

猜测第八个单词为about，猜测B g

猜测第十个单词为after，猜测F b

再次重复命令，得到



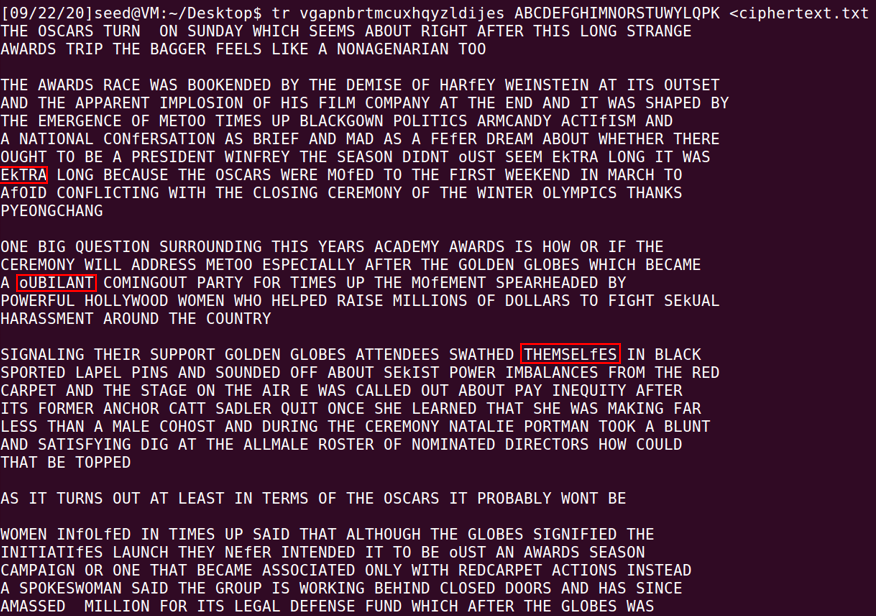
猜测第十二个单词为long，猜测L i

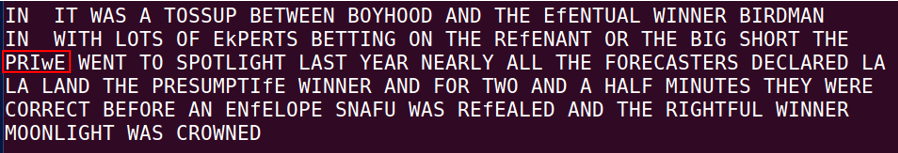
猜测第三段第三个单词为question，猜测Q j

猜测第三段第二行第五个单词为especially，猜测P e

猜测第一段文本iIsE为like，猜测K s

重复命令，得到如下结果

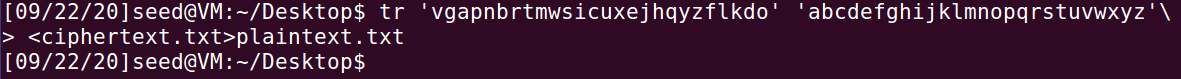


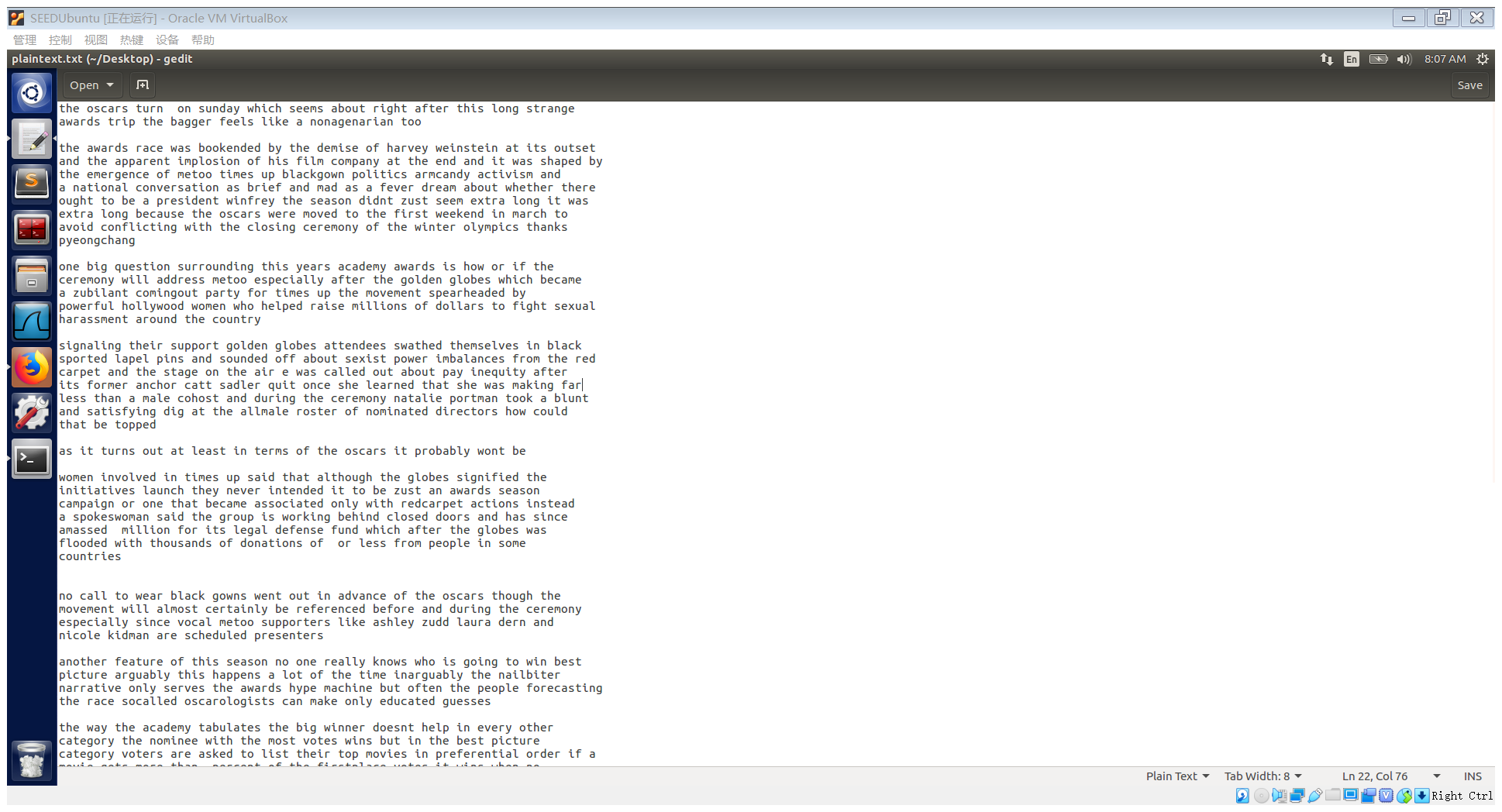


根据以上标注，可以得到X k，Z o，V f，J w

综上可以得到密码表key=vgapnbrtmwsicuxejhquzflkdo

输入命令tr ‘vgapnbrtmwsicuxejhquzflkdo’ ‘abcdefghijklmnopqrstuvwxyz’ \ <ciphertext.txt> plaintext.txt





得到明文如上

**2.2 Task 2**

本实验中我们采用穷举法，穷举key的每一位的值以及能将内容解码为可见字符的所有值。

在ubuntu中新建vigenere.py文件，输入以下代码：

def findindexkey(subarr):#该函数可以找出将密文subarr解密成可见字符的所有可能值

visiable\_chars=[]#可见字符

for x in range(32,126):

visiable\_chars.append(chr(x))

#print(vi)

test\_keys=[]#用于测试密钥

ans\_keys=[]#用于结果的返回

for x in range(0x00,0xFF):# 枚举密钥里所有的值

test\_keys.append(x)

ans\_keys.append(x)

for i in test\_keys:#对于0x00~0xFF里的每一个数i和subarr里的每个值s异或

for s in subarr:

if chr(s^i) not in visiable\_chars:#用i解密s，如果解密后明文不是可见字符，说明i不是密钥

ans\_keys.remove(i)#去掉ans\_keys里测试失败的密钥

break

#print(ans\_keys)

return ans\_keys

strmi='F96DE8C227A259C87EE1DA2AED57C93FE5DA36ED4EC87EF2C63AAE5B9A7EFFD673BE4ACF7BE8923C\

AB1ECE7AF2DA3DA44FCF7AE29235A24C963FF0DF3CA3599A70E5DA36BF1ECE77F8DC34BE129A6CF4D126BF\

5B9A7CFEDF3EB850D37CF0C63AA2509A76FF9227A55B9A6FE3D720A850D97AB1DD35ED5FCE6BF0D138A84C\

C931B1F121B44ECE70F6C032BD56C33FF9D320ED5CDF7AFF9226BE5BDE3FF7DD21ED56CF71F5C036A94D96\

3FF8D473A351CE3FE5DA3CB84DDB71F5C17FED51DC3FE8D732BF4D963FF3C727ED4AC87EF5DB27A451D47E\

FD9230BF47CA6BFEC12ABE4ADF72E29224A84CDF3FF5D720A459D47AF59232A35A9A7AE7D33FB85FCE7AF5\

923AA31EDB3FF7D33ABF52C33FF0D673A551D93FFCD33DA35BC831B1F43CBF1EDF67F0DF23A15B963FE5DA\

36ED68D378F4DC36BF5B9A7AFFD121B44ECE76FEDC73BE5DD27AFCD773BA5FC93FE5DA3CB859D26BB1C63C\

ED5CDF3FE2D730B84CDF3FF7DD21ED5ADF7CF0D636BE1EDB79E5D721ED57CE3FE6D320ED57D469F4DC27A8\

5A963FF3C727ED49DF3FFFDD24ED55D470E69E73AC50DE3FE5DA3ABE1EDF67F4C030A44DDF3FF5D73EA250\

C96BE3D327A84D963FE5DA32B91ED36BB1D132A31ED87AB1D021A255DF71B1C436BF479A7AF0C13AA14794'

arr=[]#密文，每个元素为字符的ascii码

for x in range(0,len(strmi),2):

arr.append(int(strmi[x:2+x],16))

for keylen in range(1,14):#枚举密钥的长度1~14

for index in range(0,keylen):#对密钥里的第index个进行测试

subarr=arr[index::keylen]#每隔keylen长度提取密文的内容，提取出来的内容都被密文的第index个加密

ans\_keys=findindexkey(subarr)#找出密钥中第index个的可能的值

print('keylen=',keylen,'index=',index,'keys=',ans\_keys)

if ans\_keys:#如果密钥第index个有可能存在，尝试用密钥的index个去解密文

ch=[]

for x in ans\_keys:

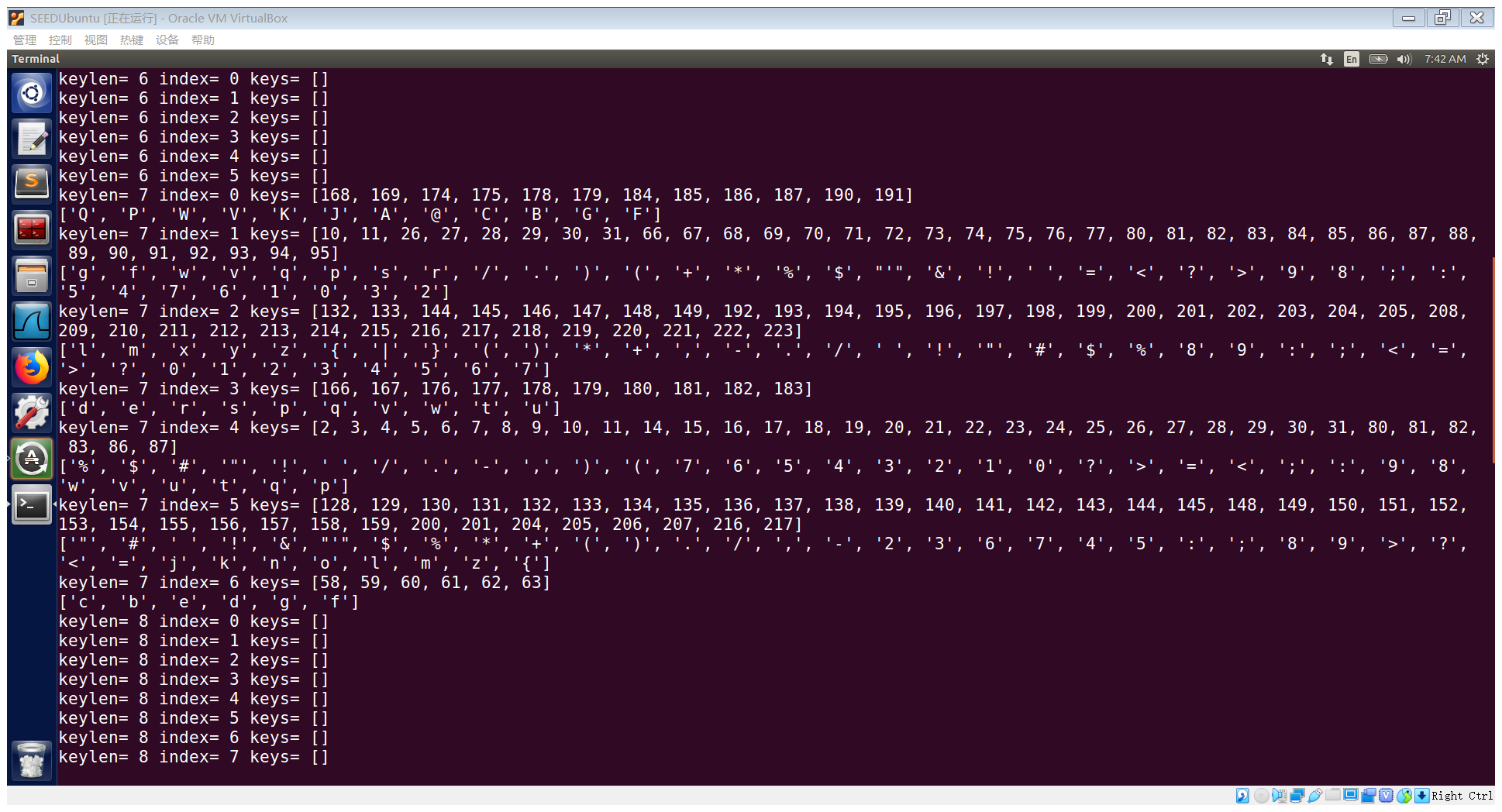
ch.append(chr(x^subarr[0]))

print(ch)

使用以下命令编译运行文件



得到结果如下：



大致可以得到密钥长度为7

若缩小范围，仅对字母、数字、空格和标点进行穷举，修改vigenere.py的内容为以下：

def findindexkey(subarr):#该函数可以找出将密文subarr解密成可见字符的所有可能值

visiable\_chars=[]#可见字符

for x in range(32,126):

visiable\_chars.append(chr(x))

#print(vi)

test\_keys=[]#用于测试密钥

ans\_keys=[]#用于结果的返回

for x in range(0x00,0xFF):# 枚举密钥里所有的值

test\_keys.append(x)

ans\_keys.append(x)

for i in test\_keys:#对于0x00~0xFF里的每一个数i和subarr里的每个值s异或

for s in subarr:

if chr(s^i) not in visiable\_chars:#用i解密s，如果解密后明文不是可见字符，说明i不是密钥

ans\_keys.remove(i)#去掉ans\_keys里测试失败的密钥

break

#print(ans\_keys)

return ans\_keys

strmi='F96DE8C227A259C87EE1DA2AED57C93FE5DA36ED4EC87EF2C63AAE5B9A7EFFD673BE4ACF7BE8923C\

AB1ECE7AF2DA3DA44FCF7AE29235A24C963FF0DF3CA3599A70E5DA36BF1ECE77F8DC34BE129A6CF4D126BF\

5B9A7CFEDF3EB850D37CF0C63AA2509A76FF9227A55B9A6FE3D720A850D97AB1DD35ED5FCE6BF0D138A84C\

C931B1F121B44ECE70F6C032BD56C33FF9D320ED5CDF7AFF9226BE5BDE3FF7DD21ED56CF71F5C036A94D96\

3FF8D473A351CE3FE5DA3CB84DDB71F5C17FED51DC3FE8D732BF4D963FF3C727ED4AC87EF5DB27A451D47E\

FD9230BF47CA6BFEC12ABE4ADF72E29224A84CDF3FF5D720A459D47AF59232A35A9A7AE7D33FB85FCE7AF5\

923AA31EDB3FF7D33ABF52C33FF0D673A551D93FFCD33DA35BC831B1F43CBF1EDF67F0DF23A15B963FE5DA\

36ED68D378F4DC36BF5B9A7AFFD121B44ECE76FEDC73BE5DD27AFCD773BA5FC93FE5DA3CB859D26BB1C63C\

ED5CDF3FE2D730B84CDF3FF7DD21ED5ADF7CF0D636BE1EDB79E5D721ED57CE3FE6D320ED57D469F4DC27A8\

5A963FF3C727ED49DF3FFFDD24ED55D470E69E73AC50DE3FE5DA3ABE1EDF67F4C030A44DDF3FF5D73EA250\

C96BE3D327A84D963FE5DA32B91ED36BB1D132A31ED87AB1D021A255DF71B1C436BF479A7AF0C13AA14794'

arr=[]#密文，每个元素为字符的ascii码

for x in range(0,len(strmi),2):

arr.append(int(strmi[x:2+x],16))

for keylen in range(1,14):#枚举密钥的长度1~14

for index in range(0,keylen):#对密钥里的第index个进行测试

subarr=arr[index::keylen]#每隔keylen长度提取密文的内容，提取出来的内容都被密文的第index个加密

ans\_keys=findindexkey(subarr)#找出密钥中第index个的可能的值

#print('keylen=',keylen,'index=',index,'keys=',ans\_keys)

if ans\_keys:#如果密钥第index个有可能存在，尝试用密钥的index个去解密文

ch=[]

for x in ans\_keys:

ch.append(chr(x^subarr[0]))

print(ch)

print('###############')

import string

def findindexkey2(subarr):#再造一个函数筛选密钥

test\_chars=string.ascii\_letters+string.digits+','+'.'+' '#将检查的字符改为英文+数字+逗号+句号+空格

#print(test\_chars)

test\_keys=[]#用于测试密钥

ans\_keys=[]#用于结果的返回

for x in range(0x00,0xFF):# 枚举密钥里所有的值

test\_keys.append(x)

ans\_keys.append(x)

for i in test\_keys:#对于0x00~0xFF里的每一个数i和substr里的每个值s异或

for s in subarr:

if chr(s^i) not in test\_chars:#用i解密s，如果解密后不是英文、数字、逗号、句号、空格，说明i不是密钥

ans\_keys.remove(i)#去掉ans\_keys里测试失败的密钥

break

#print(ans\_keys)

return ans\_keys

vigenerekeys=[]

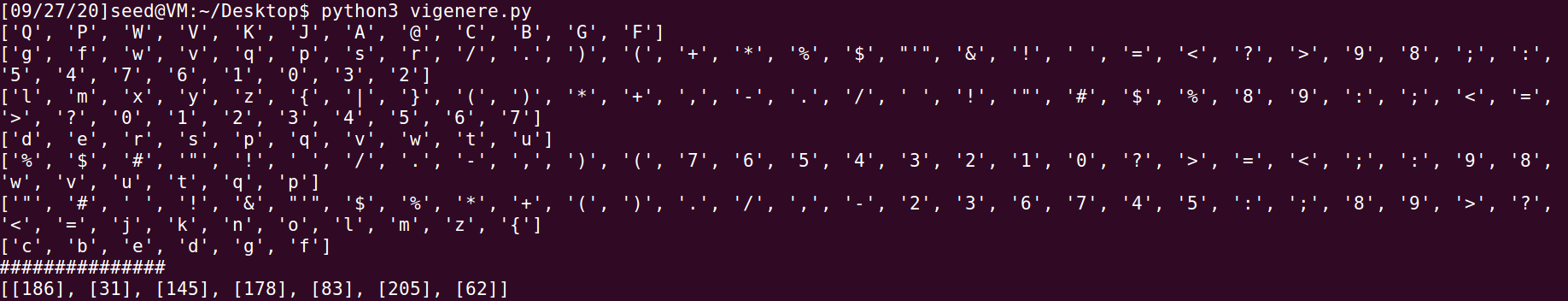
for index in range(0,7):#已经知道密钥长度是7

subarr=arr[index::7]

vigenerekeys.append(findindexkey2(subarr))

print(vigenerekeys)

可以得到：



密钥为[[186], [31], [145], [178], [83], [205], [62]]

下面使用密钥进行解密：

def findindexkey(subarr):#该函数可以找出将密文subarr解密成可见字符的所有可能值

visiable\_chars=[]#可见字符

for x in range(32,126):

visiable\_chars.append(chr(x))

#print(vi)

test\_keys=[]#用于测试密钥

ans\_keys=[]#用于结果的返回

for x in range(0x00,0xFF):# 枚举密钥里所有的值

test\_keys.append(x)

ans\_keys.append(x)

for i in test\_keys:#对于0x00~0xFF里的每一个数i和subarr里的每个值s异或

for s in subarr:

if chr(s^i) not in visiable\_chars:#用i解密s，如果解密后明文不是可见字符，说明i不是密钥

ans\_keys.remove(i)#去掉ans\_keys里测试失败的密钥

break

#print(ans\_keys)

return ans\_keys

strmi='F96DE8C227A259C87EE1DA2AED57C93FE5DA36ED4EC87EF2C63AAE5B9A7EFFD673BE4ACF7BE8923C\

AB1ECE7AF2DA3DA44FCF7AE29235A24C963FF0DF3CA3599A70E5DA36BF1ECE77F8DC34BE129A6CF4D126BF\

5B9A7CFEDF3EB850D37CF0C63AA2509A76FF9227A55B9A6FE3D720A850D97AB1DD35ED5FCE6BF0D138A84C\

C931B1F121B44ECE70F6C032BD56C33FF9D320ED5CDF7AFF9226BE5BDE3FF7DD21ED56CF71F5C036A94D96\

3FF8D473A351CE3FE5DA3CB84DDB71F5C17FED51DC3FE8D732BF4D963FF3C727ED4AC87EF5DB27A451D47E\

FD9230BF47CA6BFEC12ABE4ADF72E29224A84CDF3FF5D720A459D47AF59232A35A9A7AE7D33FB85FCE7AF5\

923AA31EDB3FF7D33ABF52C33FF0D673A551D93FFCD33DA35BC831B1F43CBF1EDF67F0DF23A15B963FE5DA\

36ED68D378F4DC36BF5B9A7AFFD121B44ECE76FEDC73BE5DD27AFCD773BA5FC93FE5DA3CB859D26BB1C63C\

ED5CDF3FE2D730B84CDF3FF7DD21ED5ADF7CF0D636BE1EDB79E5D721ED57CE3FE6D320ED57D469F4DC27A8\

5A963FF3C727ED49DF3FFFDD24ED55D470E69E73AC50DE3FE5DA3ABE1EDF67F4C030A44DDF3FF5D73EA250\

C96BE3D327A84D963FE5DA32B91ED36BB1D132A31ED87AB1D021A255DF71B1C436BF479A7AF0C13AA14794'

arr=[]#密文，每个元素为字符的ascii码

for x in range(0,len(strmi),2):

arr.append(int(strmi[x:2+x],16))

for keylen in range(1,14):#枚举密钥的长度1~14

for index in range(0,keylen):#对密钥里的第index个进行测试

subarr=arr[index::keylen]#每隔keylen长度提取密文的内容，提取出来的内容都被密文的第index个加密

ans\_keys=findindexkey(subarr)#找出密钥中第index个的可能的值

print('keylen=',keylen,'index=',index,'keys=',ans\_keys)

if ans\_keys:#如果密钥第index个有可能存在，尝试用密钥的index个去解密文

ch=[]

for x in ans\_keys:

ch.append(chr(x^subarr[0]))

print(ch)

#运行到这里，观察输出可以发现，密钥长度为7时有解

print('###############')

import string

def findindexkey2(subarr):#再造一个函数筛选密钥

test\_chars=string.ascii\_letters+string.digits+','+'.'+' '#将检查的字符改为英文+数字+逗号+句号+空格

#print(test\_chars)

test\_keys=[]#用于测试密钥

ans\_keys=[]#用于结果的返回

for x in range(0x00,0xFF):# 枚举密钥里所有的值

test\_keys.append(x)

ans\_keys.append(x)

for i in test\_keys:#对于0x00~0xFF里的每一个数i和substr里的每个值s异或

for s in subarr:

if chr(s^i) not in test\_chars:#用i解密s，如果解密后不是英文、数字、逗号、句号、空格，说明i不是密钥

ans\_keys.remove(i)#去掉ans\_keys里测试失败的密钥

break

#print(ans\_keys)

return ans\_keys

vigenerekeys=[]#维基尼尔密码的密钥

for index in range(0,7):#已经知道密钥长度是7

subarr=arr[index::7]

vigenerekeys.append(findindexkey2(subarr))

print(vigenerekeys)#输出的是[[186], [31], [145], [178], [83], [205], [62]].

print("#########")

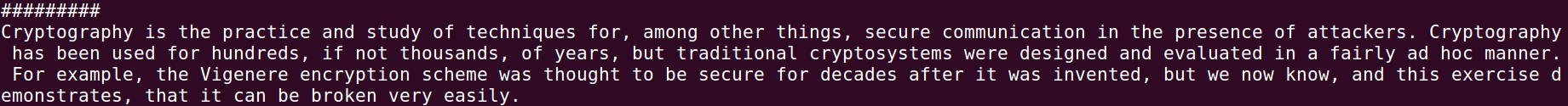
ming=''

for i in range(0,len(arr)):

ming=ming+chr(arr[i]^vigenerekeys[i%7][0])

print(ming)

得到结果：



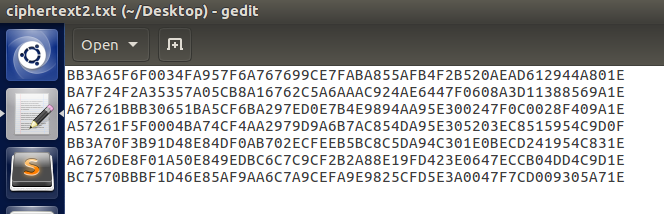
原文为：

Cryptography is the practice and study of techniques for, among other things, secure communication in the presence of attackers. Cryptography has been used for hundreds, if not thousands, of years, but traditional cryptosystems were designed and evaluated in a fairly ad hoc manner. For example, the Vigenere encryption scheme was thought to be secure for decades after it was invented, but we now know, and this exercise demonstrates, that it can be broken very easily.

**Task 3**

我们通过异或运算和ASCII码的计算规律进行破解

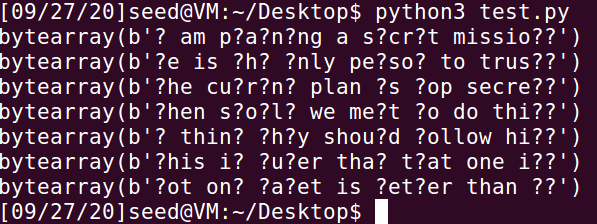
新建ciphertext.txt文件，写入密文：



新建test.py文件，并写入以下代码：



运行程序，可以得到以下结果：



可见部分明文未成功显示，我们补全后，最终可以得到以下明文：

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 I.